Escuela Politécnica Superior

Universidad Autónoma de Madrid

Master Thesis:

Low cost artificial noses with communication and collaboration capabilities.

Student: Miguel Urízar Salinas
Director: Pablo Varona

Abstract

An artificial nose sensor network has been successfully installed and used for monitoring a biodiesel plant for two years. The network is composed of different low-cost artificial noses distributed in several locations of the factory. Changing or adding noses during this time was possible, demonstrating that the system is modular and flexible. Communications stabilized by serial cables have been operating continuously without failure, as it has also made the data acquisition program except for some outages, being these external to the system. This work describes the main features found in the signals recorded by the artificial noses during loading task, leak and routine events, including inactivity periods. A set of automatic classification experiments based in support vector machine (SVM) classifiers (a generic one (SVM), a cross validation SVM (CSVM), an autoregressive kernel SVM (ARSVM) and a dynamic systems kernel SVM (DSVM)) have been developed for five different event types at two sampling rates. Different variables depending on the classifier have been sweep to find their best values: for SVM gamma, cost and tolerance and for CSVM ARSVM and DSVM only tolerance. Artificial noses have noise, drift and also history-dependent sensitivity, so the defined environment is complex. Nevertheless, for almost all events, the classifiers reach a 75% of detection or higher. The results reported in this work show that these noses can detect leak and routine events in the plant. Furthermore, continuous monitoring seem to indicate that low-cost artificial noses can be used to characterize the activity of the plant in a non-invasive manner, suggesting new uses for machine olfaction.

To my late Grandma, being a great person and even greater grandmother.
You taught me so many things...

Acknowledgements

This Master's Thesis would never be done without the inestimable help of Pablo Varona and Francisco de Borja Rodriguez guiding me through the difficulties that emerged through this two years, making easy what looked impossible. With their patience and knowledge the gave me the foundation to become a good researcher. Thanks to Ramón Huerta for granting the software needed for executing the experiments, Carlos García for allowing me to use the 3D printer for the nose housing and David Yañez for granting the artificial noses, the *OlusElda* program and making Elda plant installation. I want to thank my laboratory fellows Aarón Montero, Fernando Herrero and Jacobo Fernandez for making this phase of my life even better and helping with their support and sharing their knowledge. Special thanks to Conchi Urízar and Cecilia Martinez for helping on the document review. Also I want to thank my family for always supporting me even when my ideas have been a bit crazy. And last but not least, for the people and friends that have been there to encourage me and to help me pass through the times when the investigation seemed stagnant, many thanks.

Contents

List of Figures

Chapter 1

Introduction

1.1 Brief overview on artificial noses

In the 1920s, Zwaardemaker Hogewind discovered that adding volatile substances to water increases the spray electricity it generates [Zwaardemaker and F., 1920]. Hartman, in 1954 created an electrode that can operate with several elements covering it in order to receive different responses and, thereby, discriminating different compounds [Hartman, 1954]. In 1965 Buck studied the modulation of the contact potential to monitor aromas [Buck et al., 1965]. In 1982, Persaud and Dodd mimicked the mammalian olfactory system with an array of smart chemical sensors [Persaud and Dodd, 1982]. The term *"artificial nose"* is introduced in 1988 by Gardner and Bartlett who described it as *"an instrument which comprises an array of electronic chemical sensors with partial specificity and appropriate pattern recognition system, capable of recognizing simple or complex odors"* [Gardner and Bartlett, 1994]. The NATO Organization in 1991 had a session devoted entirely to the subject of electronic nose [Wilson and Baietto, 2009].

An electronic nose is a sensor that reacts with the chemical components that are present in the air. These elements are essential in the industry sector, specially in product quality control [Brezmes et al., 2001]. This requirement of artificial noses development arise because of the low sensibility and discriminatory capacities of human noses. They also allow to check if there are dangerous components in the environment.

Aromatic compounds have low molecular mass. They wander through the air and, thus, can be smelled. Volatility depends on the strength of the molecular bonds between them. Non-polar molecules are more volatile than polar ones. Volatile components usually have remnants of oxygen and nitrogen, and sometimes sulfur. There are natural materials that can have thousands of chemical kinds and also the aromas can change with environmental conditions.

As time has passed, the evolution of artificial noses has made them more accessible for a large variety of purposes like detecting pollution levels [Morsi, 2008], controlling the odor generated by a farm [Pan and Yang, 2009], distinguishing different brands and mixtures of coffee beans [Ulmer et al., 1997], controlling the quality of salmon [Haugen et al., 2006], predicting the storage time of a pork-meat pizza topping product [Vestergaard et al., 2007] or classifying different bacteria and human-pathogenic yeasts [Moens et al., 2006] among other hundreds of functions. The great advantage of this technology is that breaking the compounds into different molecules is not necessary, making the cost of volatile organic compounds (VOC) detection cheaper than using traditional ways of detecting chemicals.

The study of electronic noses is a very recent theoretical and practical field of study. Nowadays there are plenty of sub-fields that have not been treated or have been investigated in a superficial level. That is why, despite having a mathematical, chemical, biological, physical and psychological base, the study of artificial noses still contains many branches of research to develop and perform.

Due to its recent emergence, artificial noses normally have an expensive cost, not only because of the sensor technology used, but also because of the amount and complexity of the information generated, which has to treated to reach the same results despite of small changes in the environment. Reducing the amount of data to work with is computationally expensive and requires to discard the least useful information, not being an obvious task. This process

usually involves using data dimension reduction methods such as PCA, LDA, LLE, Kernel variations or even Fourier transforms [Huang et al., 2006]. To treat the data (once is reduced if necessary), there are many algorithms like Fourier analysis [Huang et al., 2006], neural networks [Ferreira et al., 2005], or autoregressive kernels and time series kernels [Vembu et al., 2012].

The use of new technologies for the detection of substances based in odor plumes using artificial noses represents a qualitative step forward in the protection of the safety and occupational health, establishing monitoring mechanisms and indicators to avoid the production risks, such as security measures of the workforce with high risk of exposure to different health harmful substances and an advance in the control to introduce corrective measures against the dumping of hazardous substances to the environment [Morsi, 2008, Pan and Yang, 2009]. Also, artificial noses have a large amount of applications in biomedical [Yáñez et al., 2011] and industrial scope [Stitzel et al., 2011, Dymerski et al., 2011].

The artificial noses developed so far and its related technologies have several problems. One of the most important has to do with the variation derived from the signals generated due to uncontrollable conditions, called drift [Hui et al., 2003]. To achieve a reproducible signal requires very bulky and expensive systems and, besides, a high time interval for capturing it [Depari et al., 2006]. In both cases we find the problem of low reproducibility of the signal. Therefore, the characterization of odors with these devices is complicated and expensive.

1.1.1 Low-cost artificial noses

Sometimes, different chemical noses with different odor reactions have been used in the same place to detect a wider variety of chemicals, making easier to recognize odors and even their concentration in the air [Pyk et al., 2006], making also the software to be least sophisticated in exchange of a higher cost in hardware. But as the study in electronic noses arise, the development is trying to reach simpler hardware and more elaborated software. So, as time goes on, more investigations involve low-cost artificial noses [Heilig et al., 1997, Ding et al., 2001, Depari et al., 2005, Yáñez et al., 2011]. Other good point in having low-cost noses is that the data generated is simpler than having too many sensors registering information, leading to a lower storage cost, but on the other hand, creating the need for more intelligent software [Herrero-Carrón et al., 2014].

There are some approaches to make a low-cost nose, one of them is using cheaper materials to reduce the hardware cost, other is to remove different elements form the system. The typical way is to remove the chamber, that implies removing the injection complex. Other approach is, instead of using only one high quality nose, use a set of different cheap noses that have different odor recognition capabilities. When combined, they can reach the quality of the high performance nose or even better.

To perform the capabilities of low-cost artificial noses there is another improvement that consists in positioning different noses in different places, allowing a spatial localization of the data registered [Matthes et al., 2005, Chen et al., 2004]. This improvement is good to localize odor sources in an easier way, as also for controlling wide areas with great environmental variations at the same moment, such as farms or factories.

Despite of its low cost, this kind of artificial noses have a high software expense. Less data implies less information, so the data process involved has to be more sophisticated. This makes the processing more exhaustive and complicated. Sometimes, the nose has to react in a dynamic way to the environment depending on what goal we want to achieve [Heilig et al., 1997, Kato et al., 2000].

1.1.2 Odor sensing

In twenty years there have been many advances in sensor design, materials, software, and systems micro-circuitry integration. These advances, applied to artificial noses, have given benefits to commercial industries, increasing the quality and capabilities.

The sense of smell has become essential in many industries to improve product appearance, consistency and quality. The low sensitivity and discriminatory capacity of the human nose along with its olfactory fatigue have forced the development of electronic tools that can improve the work of a human nose panel more objectively and without fatigue.

Artificial noses emulate the animal olfactory system, adding the advantage of having no fatigue. In general, electronic noses smell the mixures of organic samples as a single item rather than decomposing them, leading into a cheaper way of detecting olfactory samples. Different sensor types include metal-oxide, semiconductive polymers, electro-active conductive polymers, optic, surface acoustic wave and electrochemical gas sensors [Wilson and Baietto, 2009].

An artificial nose is composed basically of a multi-sensor array, an information processing unit, pattern recognition algorithms and a reference library [Wilson and Baietto, 2009]. The multi-sensor array is usually composed of different sensors to respond to a wider range of chemical classes. The output is attached in a distinct pattern. Identifying a simple or compound mixture is carried out without breaking up the sample. It is often used a reference library, which is built before analyzing unknown mixtures to identify compounds. This reference library is a database that stores a history of previous signals and the corresponding classification recognized by other ways. The pattern recognition algorithm is typically done with a neural network and a learning process. Thus, the differences are sought between patterns present in the reference library until a previously stipulated discrimination level is reached. The identification of unknown mixtures is performed based on the distribution of the attributes of the scents they have in common with the aromas registered in the library.

The aromas are mixtures of volatile components present in the air with a certain concentration, often called odorants or odors. Changes in the amount of odors of chemical species can be detected by artificial noses, but no changes in odorless materials. There are four measurable dimensions defined [Wilson and Baietto, 2009]:

- Threshold: Minimum concentration at which humans detect a scent. It changes from a scent to another. Aromas with low vapor pressures tend to have lower thresholds.

- Intensity: The sensation-perceived strength of a scent. Is not detected linearly, but with a sigmoidal curve. The intensity grows slowly with a

higher concentration. After prolonged exposure, intensity lowers exponentially but it recovers after eliminating the exposure.

- Quality: Is usually expressed by the use of descriptor types that describe the aroma, for instance earthy, floral, fruity, spicy, sewage, fish, Medicinal and Chemical. ASTM has classified 830 aromatic descriptors, but man fails to detect 100 of them [Ohloff, 1990].
- Hedonic assessment: satisfaction or dissatisfaction on an aroma (from 1 to 10).

This work describes the use a low-cost sensor network to monitor a biodiesel plant for detecting security events and characterizing the plant routine activities over time. In the following chapters more detailed information on artificial noses and their applications is given.

1.2 Gas Sensor types

Artificial noses have as a main component the gas sensor. This sensor is the one in charge of generating the temporal series that will be used afterwards to make different analysis. A coated material is over the sensor. This material interacts with the gas molecules in the environment by modulating an electric current. It usually measures, depending on the type of sensor, the change in the transduction, impedance, electric fields or oscillatory frequencies, the change in mass, temperature or heat generated. Other sensors measure light emitting properties of the compounds, light absorption wavelength of the color, fluorescence, polarization, etc.

Gas sensors return an analog signal that must be digitalized. Usually a simple analog-to-digital converter is used independently of the type of sensor. A generic description of each one of the most commonly used electronic sensors in artificial noses is given below:

- Metal-oxide sensors, MOS: Sensory reaction is based on an exchange of oxygen in the coating which causes attraction of electrons, thus reducing the conductivity. They consist in a ceramic tube containing a platinum coil and a doped material with small amounts of catalytic metal additives overlying it. They operate at high temperatures to make the reactions more common, so they consume a lot of energy.

- Metal-oxide semiconductor field effect transistors, MOSFET: They are based on the ability to absorb and dissolve hydrogen. The MOSFET have three layers: a semiconductor of silicone, an insulating silicon oxide and a catalytic metal. When the polar components interact with the metal, the electric field changes [Schaller et al., 1998]. It can use a thick film, which reacts only to dissociated hydrogen, or a thin one, that reacts also with hydrogen linked to molecules.

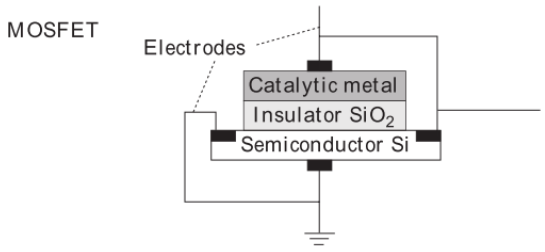


Figure 1.1: Generic MOSFET sensor and its layers.

- Conducting Polymer sensors: They suffer changes in their electric resistivity made by the absorption of gases on the surface. They are very sensitive and give a fast response, as they operate at room temperature. Usually consisting in a substrate with a pair of diodes of gold or silver and organic polymer coating, they are very susceptible to moisture in the environment.

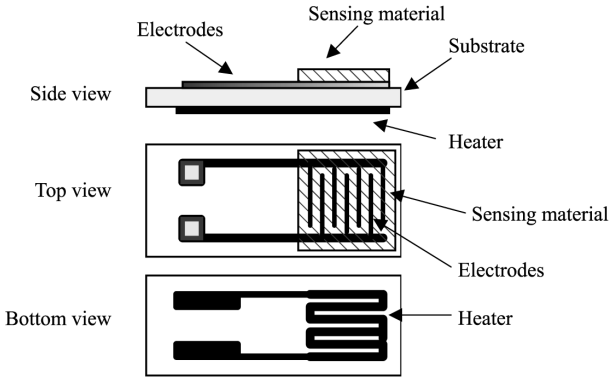


Figure 1.2: Conducting polymer sensor and its layers [Arshak et al., 2004].

- Acoustic Wave sensors: They detect acoustic waves propagated through the coating. The changes in amplitude or frequency affected by the molecular mass absorbed are measured. They are very small, cheap and sensitive to all gases. Two types exist, the BAW whose waves are on the surface, and the SAW, whose waves propagate through the substrate.

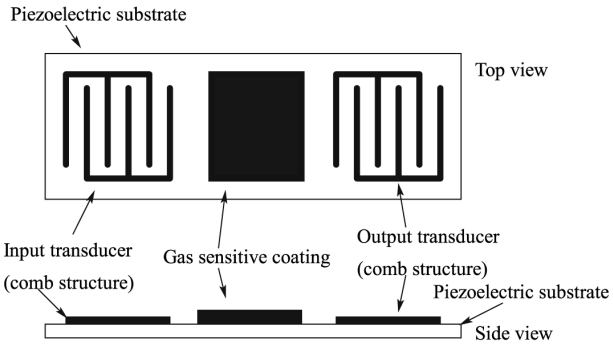


Figure 1.3: Generic SAW sensor and its layers [Arshak et al., 2004].

- Electrochemical sensors (EC): Being based on electrochemical oxidation, they are good at detecting electrochemically active gases, but not sensitive to other types [Gardner and Bartlett, 2000]. EC operate at room temperature, having little energy consumption. They are very robust, sensitive and considerably small. They consist of a layer of a precious metal coated with a hydrophobic membrane. The current flow defines the concentration of the odor in the environment.

- Calorimetric or catalytic bead: Consists in two coils of platinum wire embedded in a bead of alumina (aluminum oxide), connected to a Wheatstone bridge. One of these coils inhibits oxidation while the other boosts it. The coil is heated to combust the gas. This causes a variation in the resistance of the beads. The response is linear and takes few seconds.

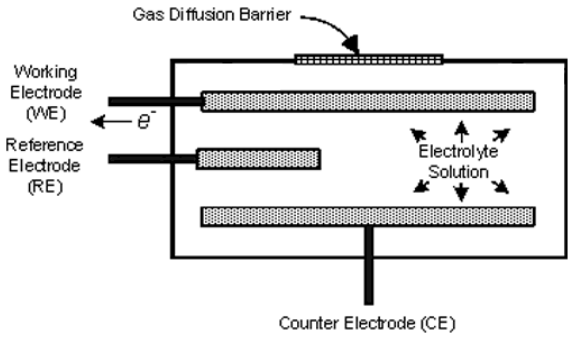


Figure 1.4: Example of an electrochemical sensor.

- Optical Sensors (OS): They measure light modulations of light sources and photo-detectors. OS detect changes in absorption, fluorescence, polarization, thickness of the optical layer, or the colorimetric dye. For detecting this last one, colorimetric sensors are used. they use thin films of chemically sensitive dyes while fluorescence sensors detect fluorescent emissions.

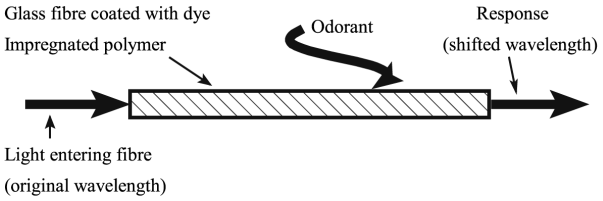


Figure 1.5: Example of an optical sensor. This one represents an optical fiber sensor [Arshak et al., 2004].

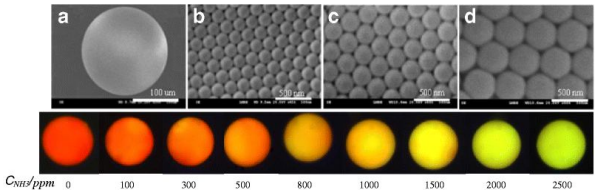


Figure 1.6: Example of an optical sensor. This one represents colloidal crystal beads. Ammonia causes the color of the sensor to change from green to red depending on the particles per million (ppm). [Xu et al., 2013].

As it is seen, different sensors have different advantages and disadvantages. For exploiting the capacities of each one, sensor arrays are used. These sensor arrays typically consist in a group of different sensor types placed in a small area to have all of them reacting to the same compound. This way, we have

different signals for the same odor that allow to analyze the environment in a more precise approach. Section 1.4 focuses in these sensor arrays.

For the development of the study presented in this document, MOS sensors are used, as being a generic purpose, cheap and low-energy consumption option.

1.3 Artificial nose components and information processing

The electronic nose is inspired in the animal olfactory system and follow the same structure, which is mainly divided into three blocks [Wilson and Baietto, 2009]:

- Sensor: This is the element responsible of capturing, usually based in electrochemical reactions (but also optical phenomena among others, see section 1.2), an analog signal that varies in function of smell and context. In animals, this would be the sensitive neurons allocated in the olfactory system.
- Preprocessing: The signal is preprocessed usually due to two reasons:
  - Noise generated by variables that affect the response of the sensor, regardless of the gases, such as drift, temperature variations and sensor saturation effects. This noise should be eliminated, because it blurs the signal [Hui et al., 2003].
  - The signal usually contains more data than necessary for the identification of gas. Different techniques are commonly used for data compressing without losing relevant information [Llobet et al., 2002].
- Categorization : Once the signal has been treated, we must know which compound has generated it. The classifications used often vary, but the neural networks are common [Wilamowski and Vodyanoy, 2008], since it is the closest classifier that animals use, the brain. SVM is also often used [Vembu et al., 2012].

So not only a gas sensor is needed for having an artificial nose. Other components are included to make the analysis of the environment possible.

1.3.1 Artificial nose components

In a first instance, the typical components necessary in a sensor are defined as the following ones [Gardner and Bartlett, 1994]:

- A sensor array, usually electrical. This is the most important part of the electronic nose. In the previous section a more in depth vision is given.
- A chamber where the sensor array is placed and where the aromatic scent is injected for a fixed time, usually by a syringe. When the mixture is extracted, the chamber is cleansed with a reference gas until the sensors are stabilized.
- An aroma delivery system that controls the injection of the aroma into the chamber. Normally this process is made by syringes. This system can also inject the reference gas to the chamber, so usually more than one syringe is connected to the aroma delivery system. Noses react more abruptly or lightly depending on the concentration of the gas. Thus, the system is usually controlled electronically so a particular concentration in the chamber is achieved.

- An electronic transistor that converts the chemical signal into an electrical one.
- An analogical-digital converter that digitalizes the signal.
- A computer microprocessor that processes the output of the signal. This component has two tasks: the first one is to reduce the output data and the second is to extract the results. The output has to be stored and also analyzed, but many times huge amounts of data are generated, so it is fundamental to make a dimensional reduction. In this way, less storage is needed and also the analysis process is faster. If the nose has to analyze in real time, the analysis has to be even faster. It is necessary to remove the redundant and useless part of the data, a work that is not trivial. Different dimensional reductions can be applied, such as principal component analysis (PCA) or canonical discriminant analysis (CDA). To extract the results, different methodologies are used, such as artificial neural network (ANN) or cluster analysis (CA) [Wilson and Baietto, 2009]. Other extra task the computer microprocessor has, is to reduce the drifting effect to the minimum. In the next section a small break down of this component is given.

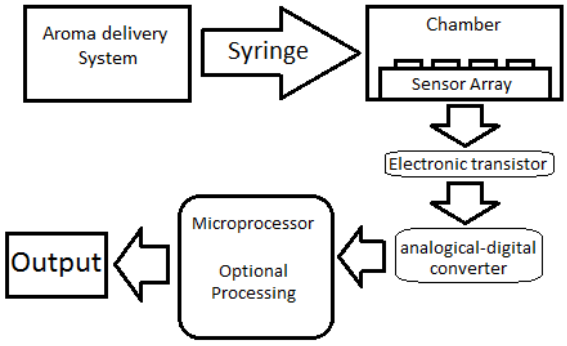


Figure 1.7: Generic work flow of artificial noses.

The low-cost artificial noses, to reach their cheap price objective, usually have neither chamber nor aroma delivery system (in some cases they have a fan to control the scent flow). They also lack the reference gas and cannot have the time lapse to stabilize the signal because of not having an isolated place to not be exposed to environmental scents. Another element that they usually have, and is not recorded as a typical electronic nose component, is a resistance that controls the temperature of the sensor. This part of the circuit allows to develop and introduce adaptive approaches as well as monitoring the temperature to have an added dimension in our output.

1.3.2 Information processing

The computer microprocessor processes and analyzes the data. There are various kinds of data analysis for artificial noses and their selection depends on the

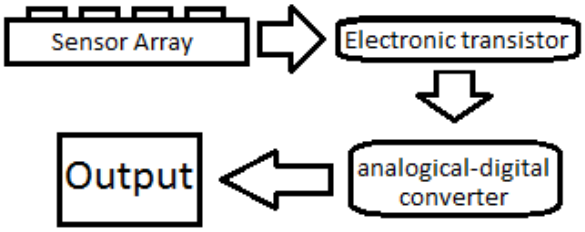


Figure 1.8: Work flow of the low-cost artificial nose used for this master thesis.

data and the type of information we want to find. The types of data analysis are divided into three categories:

- Graphic analysis: Histograms, plots, profiles, etc. They are useful for comparing samples with known sources and having a visual, easy to understand, view of the data registered.
- Multivariate data analysis (MDA). As said before, the data captured by artificial noses is usually huge, that makes necessary to reduce the dimensionality trying to remove only redundant information. It includes principal component analysis (PCA) , canonical discriminant analysis (CDA) and cluster analysis (CA). All of them reduce the dimensions of the problem when the variables are correlated. There are two types of techniques:
  - Trained MDAs: a reference library is generated for the properties and classification of unknown samples. Then is applied over the new data generated. CA is an example of a trained MDA.
  - Untrained MDAs: The database is not created. This approach is useful to compare unknown samples and when hidden relationships between them exist. PCA is useful for this case.
- Analysis of neural networks (ANN) mimics the cognitive process of the brain and contains parallel processing algorithms [Wilamowski and Vodyanoy, 2008]. Requires training and therefore a training database. The result is a success rate of each of the types referenced in the reference library.

Electronic noses are a new paradigm that requires high data storage and/or high processing cost. Sometimes, the processing cost can be reduced by arising the data storage and vice versa. For example, a reduction in dimensionality reduces de data output of the noses but has a high preprocessing cost. This approach can be useful if the analysis can be done time after the sample is captured. An example is the detection of maturity or quality of food. But for detecting hazardous odors in air is not useful, because the classification has to be given immediately, so the preprocessing and time costs have to be minimal.

New methods are developed to have more accurate responses with the same sensor, reducing costs. They require active reaction from the nose to the environment. Those methods are called adaptive methods.

Adaptive Methods

The parameters of an artificial nose can be adjusted in response to environmental or previous-history signal factors. By setting these values in a signal-dependent manner, we can achieve a significant increase in the capacity of the sensors getting better classifications and detections. Two major groups of adaptive differential variables can be defined: hardware and software [Gutierrez-Osuna and Hierlemann, 2010], but most of the time a coordination is needed in both groups.

- Hardware: Usually a predefined characteristic is monitored for each sensor. This characteristic is referenced as a single dimension in a n-dimensional space. Each of those is the individual measure of the heuristic properties. A single sensor gives very limited characteristics, that is why, to increase the features, an array of sensors is used. But other option to arise these features is to modify the transducer geometry. The modification has to be possible with the sensor working continuously. Different modifications are:
  - Temperature [Lee, 1999, Vergara et al., 2005]: The gas sensors respond to odors differently at different temperatures. The temperature of a single sensor can be varied in different ways. Rapid changes in temperature can generate many virtual sensors that react differently for the same odor. Temperature base lines can be used (sinusoidal, linear, rectangular, etc.), or sudden changes in it (in a matter of milliseconds) also can give a singular response from the sensor. Using them, individually or mixed, generate different interactions with the same gas sensor. Recently a closed loop temperature modulation method has been developed [Herrero-Carrón et al., 2014].
  - Voltage and amperage [Gutierrez-Osuna and Hierlemann, 2010]: The current is measured while the potential is modified to a fixed value as a time dependent function with respect to a known (linear, cyclic, closed circuit ...) pattern. Thus reactions with the aroma can be recorded in different ranges of reaction. This method gives a linear relationship. Amperometry is a particular example of an adaptive method depending in voltage [Buttner et al., 1990]. This adaptive method is usually combined with a chemical analysis of the aroma in liquid or gaseous state.
  - Frequency or wave length [Alause et al., 1997]: A pair of mirrors are separated by a variable space. There are peaks in the resonance energy when the space between mirrors is equal to multiples of half the wavelength of the incident light. In this way the distance between mirrors can be regulated to select a wavelength. It usually works in the infrared region. The light source is usually a diode or lamp. Quantum cascade lasers are a very adaptable example to be used as they have a grid as last mirror in the laser barrel.
  - Catalysts [Vaihinger et al., 1991]: They are heated to different temperatures to decompose or promote the chemical reactions of the sample. The sensor response patterns vary according to the type of sensor and temperature.

- Range exposure [Kummer et al., 2006]: At exposure to large intervals, all samples tend to reach an absorption equilibrium and therefore a maximum in the signal amplitude, but for short intervals, the sensor saturates only for high spreading odors. Depending on the saturation speed, can be achieved distinction between two very different samples that, otherwise, would not be distinguishable.
- Preconcentration [Pillonel et al., 2002]: When the sample concentration is below the detection limit, the sample can be enriched in an absorbent matrix. The gas is stored for a time before being released quickly using a heat pulse. The time of the enrichment phase, the matrix material absorption and desorption temperature program can be modified.
- Calibration-vapor [Gutierrez-Osuna and Hierlemann, 2010]: Often a calibration-vapor source is added to prevent external factors such as aging or drift in these olfactory systems. The kind of calibration-vapor, concentration and time exposure can be modified.
- Separation column [Reston and Kolesar, 1994]: These are columns of miniaturized gas-chromatographic units mounted in spiral shape on a silicon substrate. This allows to separate the compounds. You can change the material of the stationary phase, the temperature and air flow rate.
- Flow modulation [El Barbri et al., 2008]: In a flow pump is used a signal (square, sinusoidal, etc.). The sensor reacts with a characteristic curve when the odor flow that reaches the sensor is controlled by a defined function. After a few periods of modulation, the signal pattern becomes very reproducible. This is because the frequency encourages different patterns due to the non equilibrium regime generated. Success of this method is higher than systems that don't control the air flow. This technique records high rate of success in different compounds with a single sensor.

- Software: Some modifications can be made in the adaptation system of the signal receptor and in the classification system. Three different categories arise: adaptive filters, adaptive classification and active sensing:

  - Adaptive filters: These filters are focused in detecting non typical events (sensor failures, drifting, etc.) and delete punctual interferences called artifacts. By reacting to system or environment changes by closed loop (the system reacts to its own signal in a dynamic way giving to it some feedback), they can detect those artifacts to their posterior removal or analysis. For these purpose different approaches exist [Gutierrez-Osuna and Hierlemann, 2010]:

    * Knowledge-based approach: If the failures are well known and easily representative and distinguishable they can be defined. This way, a model of the correct working of the system and the known errors can be built. The problem with this method is that is only useful when designing the models is possible and the failure of the system is very well defined.

    * Analytic redundancy: Errors can be detected looking at redundancies, such as correlations. An example is the Evolving Window Factor Analysis (EWFA), which models the covariance in a fixed sized window. Then, events can be detected monitoring the eigenvalues of its principal component analysis (PCA) [Keller and Massart, 1991]. Recursive dynamic principal component analysis (RDPCA) correlates not only in terms of space but also time using residual error instead of the eigenvalues which speeds up the process [Perera et al., 2006].
    * Extract the baseline: Environmental interferences and chemical backgrounds are present in artificial noses, so to compensate them, the baseline must be defined. This is achieved by modeling the sensor response as a interference function with an static regression model [Sohn et al., 2008]. When more than a sensor is present in the system, the least-mean-squares algorithm gives better results [Perera et al., 2002]. The goal of using least-mean-squares algorithm is to reduce to minimum the residual error adapting it to the property changes of the sensor.
    * Blind source separation: The samples given by the sensor are modeled as a weighted sum of independent sources, so different signals are given depending of its origin without learning. This can be achieved using independent component analysis (ICA) [Di Natale et al., 2002, Wei et al., 2004, Bermejo, 2006].
    * Clustering: This method consists in defining different clusters (groups of points near each other in a n-dimensional space) with a training set. All the sample space is divided into a set of clusters. This way when a new sample is defined, a point is "painted" in the n-dimensional space and, having the clusters already defined, testing to which cluster the point belongs, the class of the new object is given. Different approaches exist for clustering a classification problem: defining only one cluster center per class [Holmberg et al., 1996], a single self-organizing map (SOM) for all the classes [Davide et al., 1994, Marco et al., 1998] or a separate SOM per class [Zuppa, 2004]. This method is useful to avoid classification problems because of drifting, but requires the correct classification of the training set and also updating the model to adapt it to the drifting. The problem comes when updating the model: usually the same classifications made by the cluster method are used as training set for creating the new cluster, which can end in classification errors because of not having checked if the previous classification was correct. A reduction in dimensionality is very advisable when working with clustering.

- Adaptive classification: These classification methods consist in learning constantly and adapting the classifier to the latest data received. Below, some models are exposed:
    * Learn++ [Polikar et al., 2002]: An ensemble of classifiers is built according to sequential information arrival. It is based in AdaBoost, a machine-learning method that builds strong classifiers by forming ensembles of weak learners (a weak learner is a learner

that is has very light improvement regarding to a random classifier) [Freund and Schapire, 1995]. Each classifier in the ensemble learns a different decision boundary. A weight is given to each sample depending in the difficulty of classifying. That way, harder samples are more probable of being selected. This classifier can support sample types that are not defined, being defined when the next classifier is built.

  - K neighbours rule [Alippi and Roveri, 2008a,b]: It consists in adding new examples to the classifier and deleting the obsolete ones. It is composed of three steps: Adapting k value, detecting non stationary behavior and removing old values once a non-stationary episode has been detected. This method requires the data to have the correct labels.
  - Adaptive resonance theory (ART) [Carpenter and Grossberg, 2009, Shukla et al., 1998, Distante et al., 2000]: This theory appears with the need of adapting a model to environmental changes without forgetting previously learned patterns. It is based in leader-follower clustering. Each new patron is related with nearest cluster and, or the cluster is updated or a new one is created. It is a method specialized in drift compensation.
  - Fuzzy ARTMAP [Llobet et al., 1999]:Based in ART, is faster than other connectionist algorithms, such as multilayer perceptrons, but it is sensitive to statistical overlap between the classes, which may force the system to create too many categories.
  - Mixtures of experts (MOE) [Kurnik, 1999]: It consists in an artificial neuronal network (ANN) in which individual classifiers are trained and their predictions are combined linearly. The weights in a MOE change continuously depending on the classifier success rate.
  - Continuous Restricted Boltzmann Machine (CRBM) [Tang et al., 2004]: It allows binary classification when high drift is present in the system. The CRBM is an ANN of three layers: an input layer with one neuron per sensor, a hidden layer that contains the structure of the classifier and an external layer that offers the class labels of the samples. The classifier is trained in two phases: the first one trains the hidden layer in an unsupervised way while, in the second phase, the output layer is trained in a supervised way. After this, the weights are frozen except for a bias for each layer so the system can stand some drift in the sensor.

- Active sensing: In animals, the olfactory system is not passive, but the system reacts in an active way to different stimulus controlling the organs to extract more specific data for the context that knows. In artificial sensing we can define also an active sensing. Depending on the input we have, the system modifies itself trying to extract more specific data depending on the signal is receiving. Some examples of active sensing are defined below:
  - Integrated preprocessing and detection (ISP) [Medendorp and Lodder, 2005]: This approach consists in making a part of the

necessary calculus as the instruments take measurements, so, reducing the data processing afterwards, part of the data can be removed to not have too much information for processing. Analogical filters can reduce the data too.

* Active odor blending [Nakamoto et al., 1995]: This consists in reproducing an odor blend by creating a mixture from its individual components. An algorithm controls the mixture ratio so the sensor response is similar to the odor blend. The algorithm gives the odors and its parts per million captured by the sensor.
* Optimize the temperature profile [Gosangi and Gutierrez-Osuna, 2010]: Some sensors give the same response to an odor M_1 at a temperature T_1 and to other odor M_2 at a temperature T_2 or even with same temperatures. So some samples can be classified in M known categories with T different temperatures. The logical approach to this problem is to measure the sensor response in T temperatures and, with the values vector, identify the class by a pattern recognition algorithm. But there is a more efficient way that consists in defining actions where each action corresponds to the capabilities of the sensor in each temperature T_m. Then, a dynamically optimized action sequence is defined to, touring the fewest number of temperatures, define the odor $M_{n,Ji2007}$.
* Hyperspectral imaging [Gehm and Kinast, 2008]: The idea is to pair a partial chemical spectrum with an already known one or a mixture of some of them. The active sensing comes into action when there is an underexposure or an overexposure making the spectrum to not display anything relevant for classification. In those cases, the system exposes more or less the odor to have the correct exposure to have a clear spectrum [Nayar and Branzoi, 2003, Christensen et al., 2002].
* Optical interference filter: They are filters designed for having a transmitter spectrum that rebuilds the regression vector, such as identifying the organic materials combination which its spectrum summed gives as result the regression vector [Haibach and Myrick, 2004].
* Statistical pattern-recognition method [Priebe et al., 2004]: It consists in building a decision tree that partitions sample space hierarchically: Nodes close to the root of the tree select basic features, whereas nodes at the leaves select sensors depending on their ability to discriminate examples from different classes.

Hardware and software methods can be mixed to have better results, even different software methods can be crossed to raise the performance of the analysis. The methods to use depend on the system characteristics including the context, the problem to solve, the time and processing force available, if the problem has to be solved in real time or deferred, etc. There is not a determinate formula to define which methods are best for each system, but an analysis to seek which one molds better is advisable.

1.4 Sensor Arrays and networks

The simplest artificial nose, as we have seen, consist in one sensor. These noses have a very low capacity of detecting volatile organic compounds or VOCs because of only having one dimension in each time-stamp. To detect a compound in the air is necessary to analyze the reaction curve that the sensor suffers when exposed to the VOC. This reaction curve can take much time to react depending on many factors, such as the sensor sensitivity, the parts per million of the compound in the air, the quantity of mixtures present or the method to analyze data, including adaptive models. Also, if not enough time has passed, the identification of the compound decreases, because these reaction curves are not lineal and also depend on the saturation properties of the sensor. Different sensors have different reaction times and reaction capabilities depending on the type of the compound present. Even two different noses that share technology can react differently to the same exposure. Having seen along the chapter different ways of performing the discrimination of compounds, another approach is available, including more than one sensor in the same nose or even working with different noses. This can be called sensor arrays, nose arrays or even networks of noses. The first gas multi-sensor array appears in 1982[Persaud and Dodd, 1982], so the technology has advanced during more than thirty years.

As not being only one time series, but several of them, the storage and processing needed grow. To evade this problem, different data reduction models exposed previously can be used, like PCA or clustering. Now there are different ways to create a sensor array, dividing it into two big groups, Single nose arrays and multiple noses array:

- Single nose arrays: These arrays are defined by having all the sensors located in the same place, inside the nose, so if it has an odor chamber, all sensors would be inside it. There can be different chambers for each sensor, but each chamber has to receive the same air flow at the same time to be considered a single nose. If it does not have a chamber, all the sensors are near each other so that, even without a chamber, the same air is present. This approach makes all the sensors to receive exactly the same air with exactly the same properties and VOCs, including the same parts per million of each one.

  The advantage is that, instead of having only one dimension for each time-stamp, we have one for each sensor. Is also possible to mix the signals given by a group of sensors achieving even more dimensions than the sum of them. Two different types of single nose sensors arise:

  - Same-sensors array [Marco et al., 1998]: All the sensors inside the nose are of the same type. They can be MOS or MOSFET, etc. Usually even being same sensors, the reaction to the same air flow is a bit different, but the reaction curves have the same shape [Barbri et al., 2008]. Different doped materials can be used in the top layer to capture the VOCs in slightly different ways [Schierbaum et al., 1990].

    This approach is the simplest and the least effective, but has some advantages: when a sensor breaks, there are other similar sensors that can replace it by different approaches, making it very tough

[Fonollosa et al., 2013]. This type of nose is very resistant to drifting effects because each sensor suffers them in a different way (this makes drifting a not predictable problem). So different methods can be used to reduce drifting, such as using artificial neural networks to detect and eliminate it [Zuppa, 2004, Tang et al., 2004], using variants of PCA models [Perera et al., 2006, Hui et al., 2003], using wavelet analysis [Hui et al., 2003], etc.

- Hybrid-sensors array [Ulmer et al., 1997]: This array is characterized for having different type of sensors in the same nose. they can be gas sensors or of other type such as temperature sensors [Yáñez et al., 2011], radioactivity sensors (as the one used in this master thesis), airflow or hearing sensors [Song et al., 2011], distal sensors (acoustic, infrared, etc.) [Loutfi et al., 2008], etc. Also more than one sensor of the same type can be present in the same nose as in the same-sensors array. Having different types of sensors, the temporal series produced are completely different making the capability of analyzing different components easier.

  Some problems are present when working with hybrid sensor noses. Not all the sensors react in the same time to the samples, making the system as slow as the slowest reaction of the sensors. This makes that an in depth analysis of the response times has to be done to achieve the best results form the array. Other problem is that the value ranges can be distinct for each nose, so models that allow different value ranges for each time series are mandatory in these cases. One approach to solve this systems is to use PCA, which combines all the series in a fixed by user dimension, reducing the data output and, therefore, the storage and processing needed to analyze samples [Pardo et al., 2005]. Other approach is to use cluster analysis, which also reduces the drifting effect in the system [Li et al., 2012].

- Multiple-noses array: An array of noses can be created to achieve different objectives more easily than with a single nose. Notice the difference between an array of sensors and an array of noses: an array of sensors can be a single nose if the air detected by al the sensors is the same at the same time, but it is an array of noses if the sensors are positioned (individually or in groups) in different places or receiving different smells. Having the noses in various positions, the system can extract information from a wider environment and draw a gas map of the area.

  Each of these noses can be single-sensor or multi-sensor noses, but they are not positioned in the same place. One advantage of this approach is that, if there are enough noses, they can detect plume sources with more success rate [Cai and Levy, 2007]. Other advantage is that if the area is big and the event to detect can occur in different places, this approach simplifies the odor detection even in variable wind environments [Pan and Yang, 2009].

  Again, some issues appear. The most noticeable issue is the communication problem. The noses are not in the same place, but the data has to converge to the point where it is processed (except in the swarm model explained at the end of this section). Different ways of communication

can be used, dividing them into two big groups, by wireless and by cable. In both technologies, a communication protocol has to be defined, varying widely depending communication methods. We are not defining the different possible protocols not having relevance in this master thesis. Wireless communications can be achieved usually by bluetooth [Pyk et al., 2006] or wifi [Song et al., 2011], being the second one the most used. In cable communications, serial cable transferring is the most used method (as is the way of communication chosen for this master thesis).

As in array sensors, the array noses can be conformed by equal sensors or different ones, being array or single sensor noses. The advantages and disadvantages for each approach are exactly the same exposed for array-sensor noses. The most important fact is to have conveniently registered where the noses are placed to make a sound map of the area [Pan and Yang, 2009].

Independently of the positions of the sensors or if there are multiple noses or a single one, the arrays and networks can be defined as fixed or mobile:

- Fixed-position nose arrays [Cai and Levy, 2007]: Typically the noses are in a fixed position. This implies the nose to not be able to move. This approach is used for odor classification, odor concentration detection or odor source localization. When the aim is to detect the localization of an odor source, is essential to have a nose network to be able to print a map of the area, but, in odor concentration and odor classification detection, single noses can be used, either array-sensor or single-sensor noses. When multiple fixed noses are used, the position of each one has to be predefined, being this a fixed value.

- Mobile nose arrays [Stachniss et al., 2008]: The noses can move and their position is not fixed. Giving them movement capability may be justified for different reasons, such as investigating the odor of a wide area that is not possible with fixed noses, not having the possibility of placing enough noses in the work area or needing the nose in different not defined positions depending on the context.

  Attaching a nose to a robot is the most typical way of making the nose mobile attributes [Lilienthal and Duckett, 2004]. The robot can be controlled by the nose or can be autonomous, depending on the aim of the system. For example, if the plan is to detect an odor source moving the robot guided by the gradient of the odor, the nose must control the mobile system [Waphare et al., 2010]. Another case is that the robots roam around the defined environment being their paths independent of the results of the data. So depending on the context, the robot must adapt to the nose signal or not. Usually is better for the robot to have wireless communication instead of cable, because it allows better mobility capacities and more independence and stability.

  Other way for making a nose mobile is to have it attached to a mobile element. In this kind environment the nose has to be able to analyze the samples detected by itself or to send the data to a control center in charge of processing the time series. Usually is important to have a method to define the nose position, to make it know or inform the place where the

samples are being captured. One very useful example of this is to attach it to a human being, allowing the nose to detect hazardous odors that can put in risk the person's health. Other option is to include it into a car or other autonomous element to make a network that can allow a central system to analyze and form a map of an area. The possibilities of mobile noses are countless.

A last kind of network noses are the swarm noses. Different characteristics define these systems:

- Many noses: The structure of this system consists in having many (tens or even hundreds) of mobile noses.
- Independent work-flow: The data generated can be massive, so each nose has to resolve their own classification problem.
- Neighborhood communication: To have a better knowledge of the environment, each nose can communicate with their nearest noses, exchanging information of their resolutions. Each nose can not connect to every other nose, but only with the ones that are near it.
- Optional Central system: The points defined above make this a non hierarchical system, but a central system can be present where all the relevant data can converge to a posterior analysis of the context and the noses reactions. Even this central system can give some orders to the noses individually or collectively.

The different array and network noses can be combined as desired in any scheme to reach a better capability to resolve the intended problem, not having boundaries in the way of mixing them, giving more complex systems that can allow to solve harder problems. Having so many options makes important to analyze the problem to choose the nose structure that best fits in the context.

1.5 Overview on artificial nose applications

There are different ways to monitor environments. The most general ones are invasive and non-invasive monitoring. The invasive monitoring is the one that implies modifying the environment or extracting more data than is necessary, risking the exposition of it or allowing to be extracted by third parties. Using cameras implies invasive monitoring due to recording data that can be used incorrectly and violates privacy of the environment. Other example of invasive monitoring is a chemical analysis, since it destroys the sample. A blood test is also invasive monitoring because it extracts the blood sample used. Non-invasive monitoring does not risk any part of the environment and does not expose privacy of the context. Artificial noses are a non-invasive monitoring system because of not giving information that can be used for malicious or out-of-the-system purposes and not altering the environment in any way.

One of the most important goals for the artificial noses and their related investigations are focused in monitoring environments. The artificial noses, having the ability to detect different airborne substances and odors, are a great and cheap non-invasive method to monitor different contexts having a wide range of functionalities. These functionalities in environments can be classified in different groups:

- Classification of different odors: The idea of this functionality is to detect and classify different odors in the air surrounding the artificial nose [Chomtip, 2009]. Usually there is a learning process before the nose can be integrated in an environment, so, typically, a database is used for differentiating chemicals present in the air. A variation of this capability is to detect different concentrations of the same chemical component [Gajdo, 2005] or also combining detection and classification of different components and their concentrations [Nakata et al., 1996].

- Detection of different concentrations: With the same odor in the environment, the nose can react in different ways depending on its concentration. Usually the concentration is measured in parts per million or ppm. Ppm measure indicates the number of particles of odorant present in one million particles, so is very similar to a percentage. The reaction of the nose is different depending on the concentrations [Barbri et al., 2008] even with more than one compound mixed in the air [Nakata et al., 1996]. If the nose does not saturate, the curves represented by the time series on each concentration are similar but with different amplitude, what makes useful to use a supervised learning method. If the nose saturates, the values given for each concentration are harder to discriminate.

- Detection of odor sources: The source of an odor can also be classified by artificial noses, using different approaches for this finality, such as mobile robots [Pyk et al., 2006, Loutfi et al., 2008, Waphare et al., 2010] or using spatially-distributed electronic noses [Matthes et al., 2005]. Diffusion and advection predictions or physical models are used for simplifying this task [Pan and Yang, 2009]. The development of the detection and classification of odor plumes in real contexts is one of the branches that has been investigated the least. That is because it is hard to detect and

classify plumes in controlled surroundings and it becomes harder in environments that are not controlled. For those purposes, other sensors are included in the equation, such as temperature, humidity, air flow sensors, etc. and also different environment simulation approaches are used. Even the simulation of animal behaviors is used to reach a reliable detection and classification [Lytridis et al., 2006].

Artificial noses are being integrated into the industrial field due to their capabilities to detect different odors without interfering in the context. Those real applications are, for example, food quality (freshness [Dutta et al., 2003], characteristic odor [Lozano et al., 2007], maturity [Berna et al., 2004] ..), meat products [Barbri et al., 2008], agricultural production in plants [Gómez et al., 2008], pathologies generated by plants [Siripatrawan, 2008], product identification [Lozano et al., 2006], medical pathologies [Yáñez et al., 2011] or chemical detection [Raman et al., 2009]. This is because artificial noses are usually automated, but require data processing. Their training is easier than traditional analytical methods because they do not require reagents. The advantages of artificial noses in comparison with other methods are: they give quick results, they do not destroy the sample and they are cheaper and more portable. With these reasons, they are accessible to industries that cannot allow to have extra costs or long time expending ways to analyze the products. But on the other hand they have reproducibility problems, the recovery depends too much in the temperature and humidity and they do not identify individual chemicals, so artificial noses do not fit in every industrial context.

Another way of classifying artificial noses is in function of the activity they are performing, having three great blocks: security, health and quality control:

- Health applications: These are the activities that are in charge of helping in the medical branch. There is a connection between differences in the aroma of diseased versus healthy human odors. Dogs are already used in these applications [Wilson and Baietto, 2009, Pickel et al., 2004]. Some examples are: pathogenic microorganisms detection [Gibson et al., 1997], pneumonia diagnosis [Hanson and Thaler, 2005] or early screening for the presence of lung cancers [Machado et al., 2005].

- Security applications: These applications include detecting hazardous substances in the air, explosives, etc. The artificial noses react to different compounds present in the air, having the capability of detecting hazardous substances, even some that are odorless for human olfactory system. Most research in this field has been done in laboratories, being a pair of examples the detection of hazardous gases[Huang et al., 2006] or odor plume source detection with an environment affected by wind [Matthes et al., 2005].

- Quality control applications. These applications can go from detecting the maturity of a vegetable to controlling the production of a factory. The function of artificial noses in these environments is to give a quality measure of a product or help to improve it. Quality control applications have been emerged in food industries [Schaller et al., 1998, Berna et al., 2004, Gómez et al., 2008] or livestock farms [Pan and Yang, 2009].

In this master thesis we focus on the study of security and production events. Also we check the ability to predict their occurrences and developments through studies of the data sets received assessing their temporal order. This study is performed on a real working environment, specifically a biodiesel plant located in Elda. In chapter 2, a in depth vision of this environment is given.

Chapter 2

Methodology

2.1 Monitoring an industrial plant

For this project, artificial noses are being tested in a biodiesel processor plant. This plant is located in an industrial park at the outskirts of Elda, a city located near Alicante (Spain).

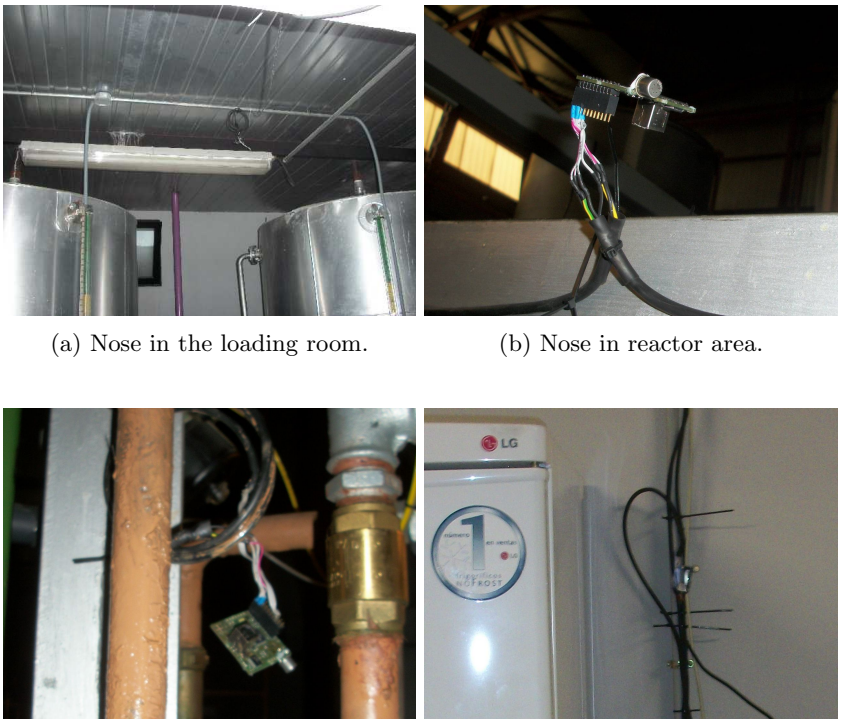
Figure 2.1: Elda biodiesel plant. Front image.

The main purpose of this plant is to generate biodiesel fuel from vegetable oils of different origin. Other goal of this plant is to recycle different used oils, so, between a 15% and a 20% of the raw material come from collection in homes and catering establishments, usually previously used for frying purposes. The rest of the oils are derived from soybeans, colza and sunflowers.

The purpose of installing the artificial nose network in this plant is to warn of possible hazardous leaks inside it. As being a plant that performs dangerous chemical reactions and also stores hazardous substances, a leak can be very noxious for the workers and the environment surrounding the area. Artificial noses can be a non-invasive approach for the detection of these leakages.

There are some emblematic points where were decided to locate the different noses:

- Load room: In this room, the different vegetable oils are mixed with methanol and caustic soda. Caustic soda is loaded in the system manually. This load is performed in direct contact with the air of the room, so the artificial nose is theoretically able to detect when soda is included in the process. Also there can be leaks in this room which are very dangerous for worker's health, so, this nose is meant to give us capacity to detect this hazardous threat. This room is separated from the rest of the factory by an isolation gate and only has one little window overlooking the courtyard. This area has the problem that the caustic soda deposits on the circuits and the gas sensor. One nose stopped working correctly after approximately half a year working. Is also important to take in account that this sediment over the nose can generate a big drift. This makes the



(a) Nose in the loading room.

(b) Nose in reactor area.

(c) Nose in centrifuge area.

(d) Nose in the control room.

Figure 2.2: Photographs of placements of artificial noses in Elda plant.

nose to react in different ways to similar events when a big lapse of time is in between them. Photograph 2.2a shows where this nose is placed.

- Reactor/reaction area: Two noses are located in between the area where the reactors are. In this place the noses are meant to capture the reactions that are being made. These reactions are long processing ones, lasting more than six hours. This area is full of oil on all surfaces, so the noses have been located on top of the containers evading direct touch with oil to prevent the noses from having a short circuit or break down. Anyways is very possible that a drift can be generated because of this direct contact with oil, bunging up the sensor. Photograph 2.2c shows how this nose is placed to evade oil contact.

- Centrifuge and sample collecting area: One nose is placed over a tap that allows to extract samples to test the quality of the biodiesel. Also is near the centrifuge, which separates different materials by centrifuging them. This nose can detect when the samples are taken and when the centrifuging process is active. Again, as in the reactor area, oil is over every surface, so, to evade it, the nose is placed without touching any surface. Photograph 2.2b shows where this nose is placed.

- Control room: This area is a small room inside the factory which is phys-

ically separated from the rest of the complex by some walls with a door and a window that communicate directly to the room where the centrifuge and reactors are, so, the air inside should have the same characteristics as in the centrifuge and reactors area. Inside this room is where computers and electric systems used to control the production of the plant are placed. It also contains a fridge and a microwave oven for the workers to have their meals. Two noses are installed in this area, one inside the room and one outside. The nose inside this place, then, is able to register the lunch times and react to a wide quantity of different stimulus. Photograph 2.2d shows that this nose is placed near the fridge and on top of the computer that captures the temporal series of the noses. The nose outside the room registers an area where the workers usually pass by and also is quite centered in the middle of the factory, so can give us general values of the plant.

The purpose of these noses installed in the Elda plant can have different approaches. The two most important and in which is focused this master thesis are controlling biodiesel production and detecting hazardous emanations:

- Hazardous odor detection: The plant makes different chemical reactions, some of them expel noxious gases that are very dangerous for human beings. A leak can be very hazardous for the factory workers and people in the surroundings. In the loading room, sodium methoxide is created mixing caustic soda and methanol. This is an exothermic reaction. Sodium methoxide is highly corrosive to skin contact and the inhalation of this vapor can be very harmful. As methanol, sodium methoxide is odorless, so almost undetectable for a worker without the proper tools. Artificial noses can be present to detect when a methanol or sodium methoxide leak appears and give backup to the actual security methods that exist. As the dangerous gases are mixed in the loading room, the leaks that are interesting to detect are placed in there, where an artificial nose has a fixed position.

- Control production: Not all the oils used in the plant come from used oil, but one of the objectives is to recycle them, so a high part of the oil used in the process is recycled. Being a recycling oil plant, Elda biodiesel factory is not working in regular intervals or with fixed timetables. When the machines involved in the process of creating biodiesel are active, the air of the environment changes and it has new odors and plumes that are not present with the same intensity when the plant is inactive. Also, when the machinery is on, the workers are too, which makes the air to move. So, the artificial noses installed in the plant, being spread throughout the complex, have the possibility to detect if the plant is active or is in a rest period waiting for new oil to be collected.

Another typical use of an artificial nose in a factory is the odor source localization. This functionality is not the objective of the artificial noses in Elda because of plenty of odors present, making it difficult to discriminate the odor that we want to detect from the others and harder to detect its source. Other reason is that when a leak appears, the source is not as relevant as the detection time, to be able to evacuate the plant as fast as possible. So, detecting leaks

in the plant is important and can raise security of the plant, but detecting the origination spot is not so relevant.

2.2 Artificial nose sensor network in Elda plant

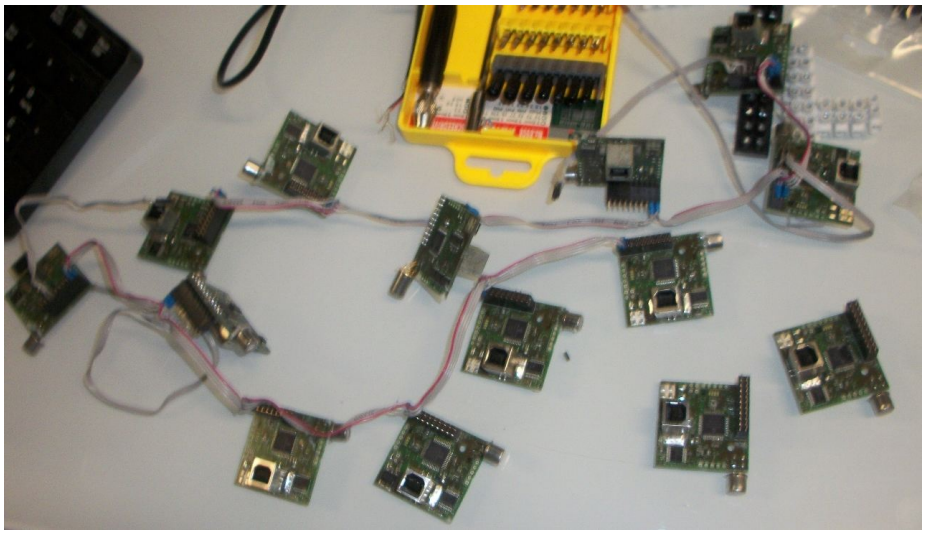


Figure 2.3: Artificial noses made by David Yañez.

Sensors for this master thesis must have the highest sensitivity to the chemical compounds that are wanted to be detected. Arrays also should have a low sensitivity to the variable parameters of the environment, to be affected the least by those not controlled parameters, particularly humidity and temperature. Also they are expected to have a fast response and recovery to detect as fast as possible changes in the environment. Finally, the sensors must be inexpensive, easy to transport and very stable.

2.2.1 The artificial noses and their sensors

In the biodiesel plant artificial MOS gas mono-sensor noses were introduced to register temporal series to analyze [Yáñez et al., 2011]. There are a total of 6 noses spread throughout the biodiesel plant, as mentioned in previous section. The noses used are made by *Deutecno noses & sensors S.L.*, a company that works side by side with the laboratory where this research has been made. They have different chemoresistive sensors, but all of them have in common a high sensitivity to low concentrations of odorants, a low drift over time, low power consumption and a high robustness. All of them increase their conductivity in function to the concentration of the odorant in de surrounding air.The sensing element is comprised of a metal oxide semiconductor layer formed on an alumina substrate of a sensing chip together with an integrated heater. This heater allow the system to integrate adaptive methods based in temperature. A simple electrical circuit converts the conductivity changes to an output signal that corresponds to the odorant. The signal generated by the sensor is not amplified, it is only conditioned with an LMC6484 amplifier in voltage follower-mode and sent to the Analog-to-Digital (A/D) converter module to be acquired and stored in a computer. The voltage range of this signal is 0–5V. Four type of MOS

sensors are installed in the Elda plant:

- TGS 2600: The TGS 2600 (Figaro Engineering Inc.) is a chemoresistive sensor that provides a good combination of high sensitivity to low concentrations of gaseous air contaminants such as hydrogen and carbon monoxide which exist in cigarette smoke. The sensor can detect hydrogen at a level of several ppm. Being of general purpose, this sensor can help us to detect these kind of compounds in the air. TGS 2600 requires a heater current of only 42mA.

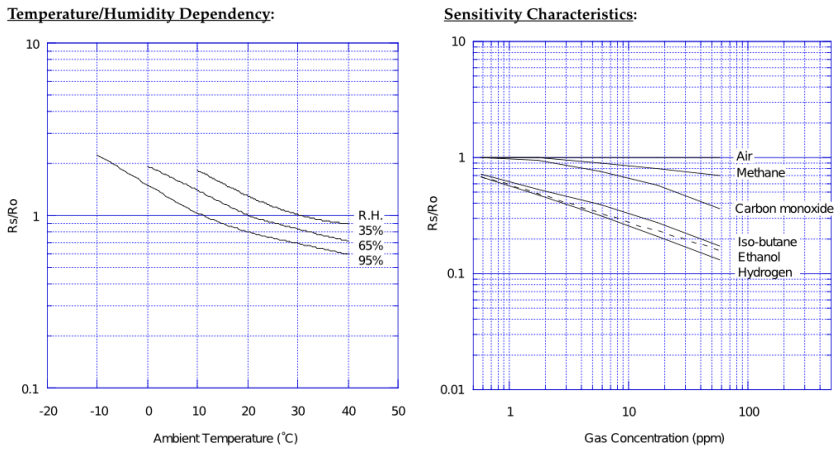


Figure 2.4: Graphs of sensor TGS 2600 output humidity and temperature dependence (figures obtained from Figaro data sheet).

Both temperature and humidity quite affect the sensor, causing necessary to measure these environmental variables. As is shown in Figure 2.4, with different humidity and temperature, on the same context, the response varies. Also two different compounds with different ppm can have the same response in the sensor, such as 40 ppm CO and 5 ppm iso-butane (Figure 2.4).

- TGS2602: The TGS 2602 (Figaro Engineering Inc.) is also a chemoresistive sensor that has high sensitivity to low concentrations of odorous gases such as ammonia and H2S generated from waste materials in office and home environments. The sensor also has a high sensitivity to low concentrations of VOCs such as toluene emitted from wood finishing and construction products. This sensor can give the system a general air quality measurement. TGS 2602 requires a heater current of only 42mA.

  This sensor is not affected too much by temperature, as we can see in the first graph of the Figure 2.4. But on the other side, different compounds with different ppm have the same response in the sensor, such as 10 ppm ethanol, 2 ppm hydrogen sulfyde, 1 ppm toluene and 30 ppm amonia. This can be seen in the second graph in Figure 2.5. This makes necessary to contrast this sensor's response with other time series.

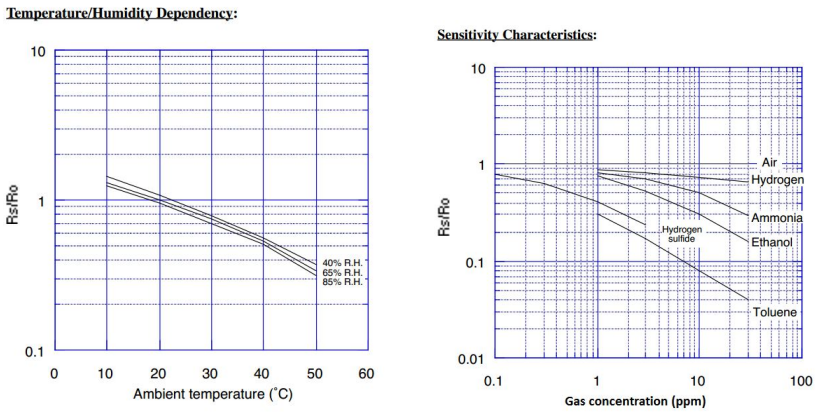


Figure 2.5: Graphs of sensor TGS 2602 output humidity and temperature dependence.

- TGS2611-C00: The TGS 2611-C00 (Figaro Engineering Inc.) chemoresistive sensor combines very high sensitivity to methane gas with low power consumption and long life. This sensor possesses small size and quick gas response, making it suitable for gas leakage checkers. TGS2611-C00 requires a heater current of only 56mA.

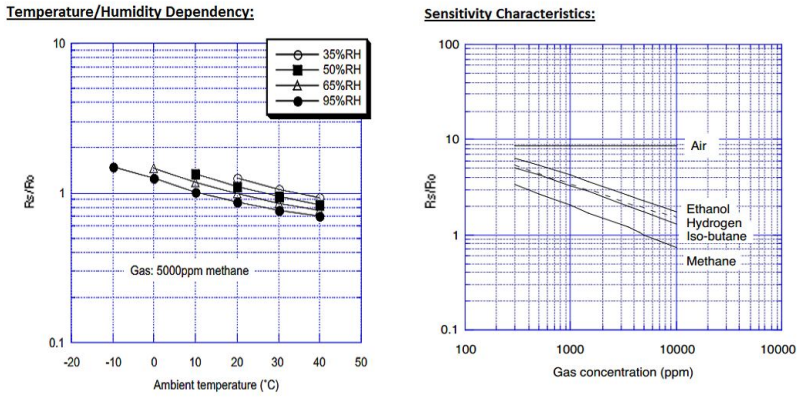


Figure 2.6: Graphs of sensor TGS 2611-C00 output humidity and temperature dependence.

This sensor is also not affected too much by temperature, as we can see in the first graph of the Figure 2.4. But on the other side, different compounds with different ppm have the same response in the sensor. The sensor response to air is almost trivial and also the representation curve of the gases to which the sensor responds are very similar, (see second graph

in Figure 2.5). So this sensor can give information of presence of a gas which it reacts but not to identify it easily.

- TGS2620: The TGS 2620 (Figaro Engineering Inc.), another chemoresistive sensor, has high sensitivity to the vapors of organic solvents as well as other volatile vapors. It also has sensitivity to a variety of combustible gases such as carbon monoxide, making it a good general purpose sensor. TGS 2620 requires a heater current of only 42mA.

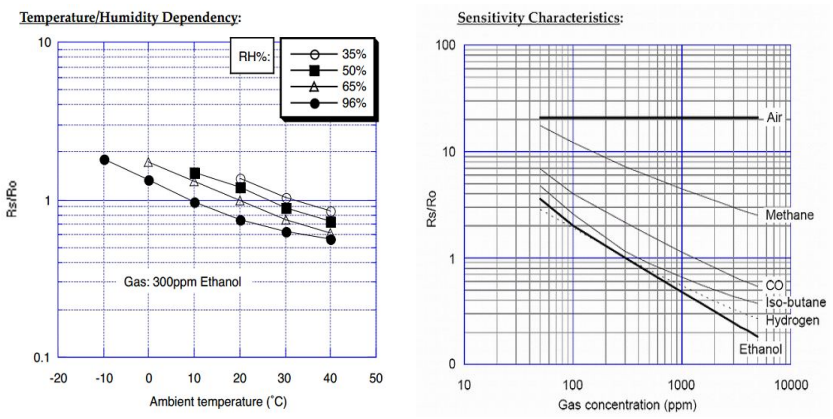


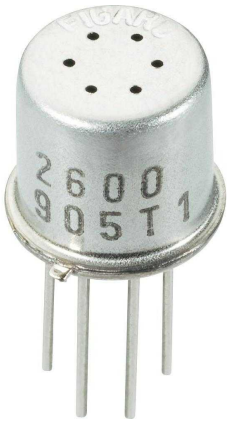
Figure 2.7: Graphs of sensor TGS 2620 output humidity and temperature dependence.

This sensor is also not affected too much by temperature, as we can see in the first graph of the Figure 2.4. Different compounds with different ppm have the same response in the sensor except for the methane, that can be identified clearly and is not near the response of the other gases. The sensor response to air is almost trivial. Response to different substance concentrations can be seen in second graph in Figure 2.5. This sensor can detect by itself very clearly the methane compound.

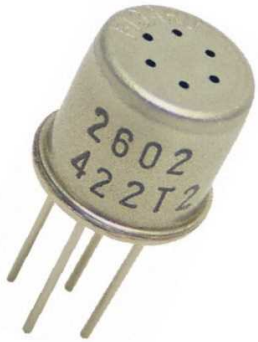
As seen in the reaction graphs of the different gas sensors, the sensibility to environmental variables from noses is very high. A way of controlling this variables or, at least, measure them is essential to simplify the analysis of the data. For that purpose, other non gas sensors are also integrated in the nose system. They are humidity, temperature and radiation sensors, which allow the system to take in account those different environmental variables that affect the gas sensors:

- Temperature sensor: The TC1047A is the temperature sensor installed. The TC1047A is a linear voltage output temperature sensor whose output is directly proportional to the measured temperature. The TC1047A can accurately measure temperature from $-40\,^{\circ}\mathrm{C}$ to $+125^{\circ}\mathrm{C}$. The power supply range is from 2.5V to 5.5V.

  The TC1047A sensor has an output voltage that varies linearly with temperature in degrees Celsius. The output voltage is 100 mV at $-40\,^{\circ}\mathrm{C}$,



(a) TGS 2600 gas sensor.



(b) TGS 2602 gas sensor.

(c) TGS 2611-C00 gas sensor.

(d) TGS 2620 gas sensor.

Figure 2.8: Photographs of the different gas sensors installed in the artificial noses.

500 mV at 0°C, 750 mV at +25 °C and 1.75V at +125 °C. The output response allows a predictable temperature measurement over a wide range (Figure 2.9 making it ideal for applications in extreme environments.

- Radiation sensor: NSL-19M51 is the radiation sensor chosen for the system. The sensor works with two cadmium sulphide (cdS) photoconductive cells with spectral responses similar to the human eye. The resistance falls with increasing light intensity. The NSL-19M51 is ceramic plastic encapsulated for moisture resistance. Light dependent resistors have a particular property in that they remember the lighting conditions in which they have been stored. This memory effect can be minimized by storing the sensor away from light presence prior to its use. As the noses are registering data twenty four hours a day, is not possible to avoid memory effect, so weird responses from the sensor can emerge. The operating temperature

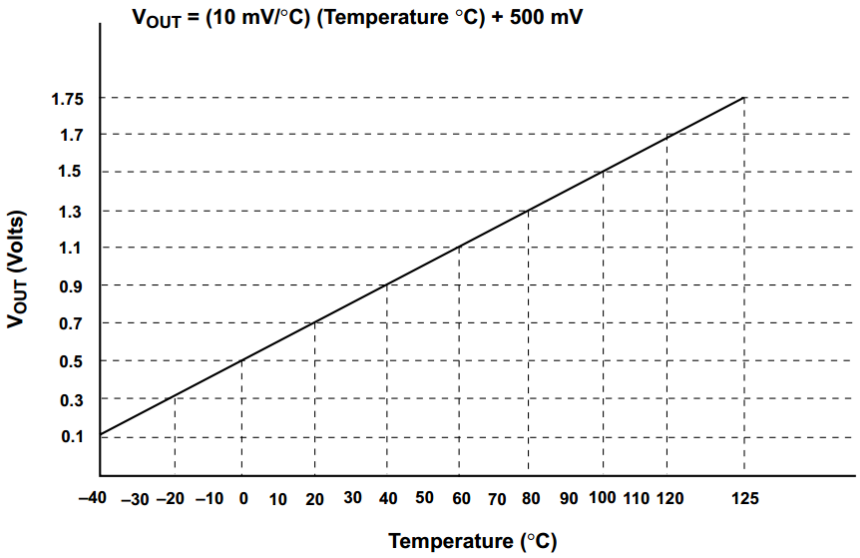


Figure 2.9: Temperature against voltage graph of TC1047A sensor.

is between $-60\,^{\circ}$C and $75\,^{\circ}$C. The power dissipation decreases linearly from 50mW at $25\,^{\circ}$C to 0W at $75\,^{\circ}$C.

- Humidity sensor: The HIH-4000-001 sensor is the chosen humidity sensor installed in some of the noses. It has a linear voltage output with direct analogical output, so an analog-to-digital converter (ADC) is necessary for registering the values of this sensor (this is achieved by the central microcontroller of the artificial nose). It has a typical current draw of only 200 μA which fits perfectly in an autonomous low consumption nose. The sensor is a laser trimmed thermoset polymer capacitive sensing element with on-chip integrated signal conditioning. Its multilayer construction provides excellent resistance to most application hazards such as wetting, dust, dirt, oils and common environmental chemicals, very useful for the context in which the noses have to work.

2.2.2 The nose sensor network

A nose sensor network has been installed because of the requirement of controlling all the area of the factory. The sensors are metal oxide semiconductor (MOS) type. In some parts of the development of this research, some electronic noses will form part of the system, not being able to classify them as multi-sensor array but a multi-nose array, because the noses are independent with different heating systems and different localizations. This way we have not only different sensor types (all metal oxide variations) but also different locations of the sensors. So the system approach is a hierarchical nose sensor network. All the noses are plugged to a central computer placed in the control room. Cable connections were used instead of wireless for some reasons. One is that this way batteries are not necessary, so the noses can work nonstop indefinitely. Other

is that there are no possibilities of information to be lost due to connection problems or lost packages. But the problem is that to install them, they have to be in fixed positions and also wiring the cable had problems, because it had to be in a safe place to not break or disconnect and to not disturb the workers.

Six noses are installed along the plant. The noses are installed as goes:

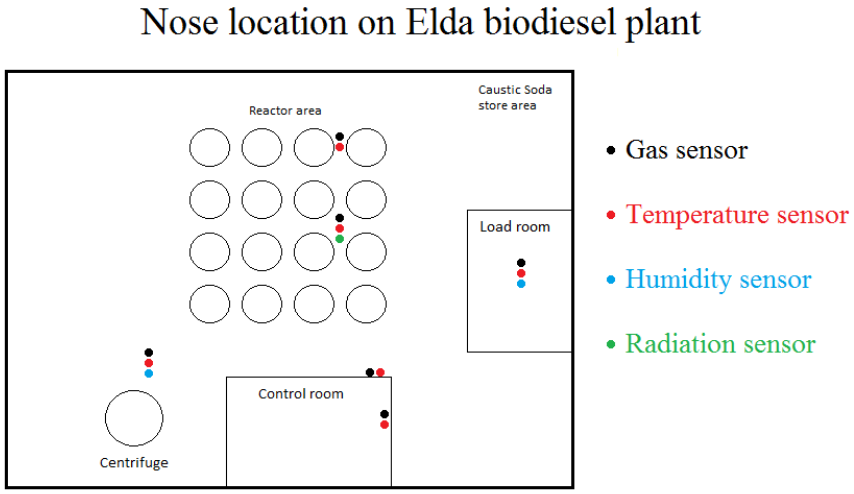


Figure 2.10: Location of the sensors in the Elda factory.

- Nose 1: Located inside the control room (where the microwave oven and fridge are), this nose is endowed with a gas sensor type 2600 (general purposes) and a temperature sensor. In this room not only control issues are executed, but also is where the workers eat and store the food. There are fifteen people (three per turn) that don't eat at the same time. We do not have the times when they eat because they are variable. The sensor is located exactly over the central system near the fridge and the microwave oven (less than two meters). Figure 2.2d shows this nose.

- Nose 2: The second nose is located in the reactor area. The sensors included in this nose are a gas type 2602 (general air quality checker) and temperature. It is located with nose three in the reactors area. In this area the impurities are removed through a process that takes hours. Because of the reactors process, there are traces of oil on all surfaces that can make the nose to saturate and stop working correctly, a fact that must be taken into account since the drift can be very high. Nose is located on top of a beam that holds the entire reactor complex, that way is less affected by the oil. Figure 2.2b shows nose's position.

- Nose 3: Third nose is also located in the reactor area with the second one. This nose has a gas sensor type 2611 (perfect for detecting gas leakages, which makes it very useful in this part of the factory), a temperature sensor and a radiation one. Being in the center area of the plant and on

top of the reactor area, radiation sensor will receive a lot of light, so it is possible that the radiation sensor saturates, but also can give the system an idea of when the lights are on or off at night time, so is a good approach to simplify the activity control of the plant. Again, this nose can suffer from high drifting because of the oil leakages of the system, that is why it is located over another beam, the same as nose two, but at four meters from it. Figure 2.2b shows nose's position.

- Nose 4: Located outside of the control room, this was the last nose to be included in the system. From this position, the nose is near the center of the factory, being in a place where it is not near any emblematic place. Having a 2611 sensor (which gives good response from gas leakages) and a temperature sensor, this nose is useful to register a general view of the air in the plant. Also is near a corridor used often by workers to go from one area of the plant to another and to enter in the control room, so the nose can register many different source VOCs.

- Nose 5: This nose is installed in the loading room. The sensors present are a 2611 type (again perfect for leakage detection), a temperature sensor and an humidity one. Being this room where caustic soda and methanol are loaded, the need of detecting a leakage in this room is essential. This was the first nose to include in the system due to its location importance. Here two tanks mix caustic soda and methanol to generate sodium hydroxide and methoxide. The way of introducing the caustic soda is manual and exposed to air, so the nose should register soda loads as they are made. Methanol is injected in the tank by a pipe, so the compound should not be present in the air and, therefore, the nose should not register it. The room is isolated from the rest of the complex, so this nose should not be exposed to the same air flow as the rest of noses. A skylight allows air renewal to clear it and not allowing the nose to become saturated. Figure 2.2a shows nose's position.

- Nose 6: The last nose is placed in the centrifuge and washing area. It has a 2620 gas sensor (general purpose), another of humidity and a temperature sensor. In this place the biodiesel is centrifuged. Also there is a tap just behind the nose to extract biodiesel samples to analyze them. Every time the tap opens the nose will be able to register it. Also being near the centrifuge, this nose will react to leaks. This area is also covered with oil, so the nose has been put on a beam without contact with any surface.

The figure 2.10 shows the position of the various sensors in a rough map of Elda plant.

To prevent waste accumulation in the sensor and circuitry of the nose, a housing is designed to cover the nose (see Figure 2.11). This housing is made by a 3D printer. The housing leaves the gas sensor exposed while the rest is covered by a plastic box, letting some holes for cables and the status lights to be seen. For more information, see appendix 2.

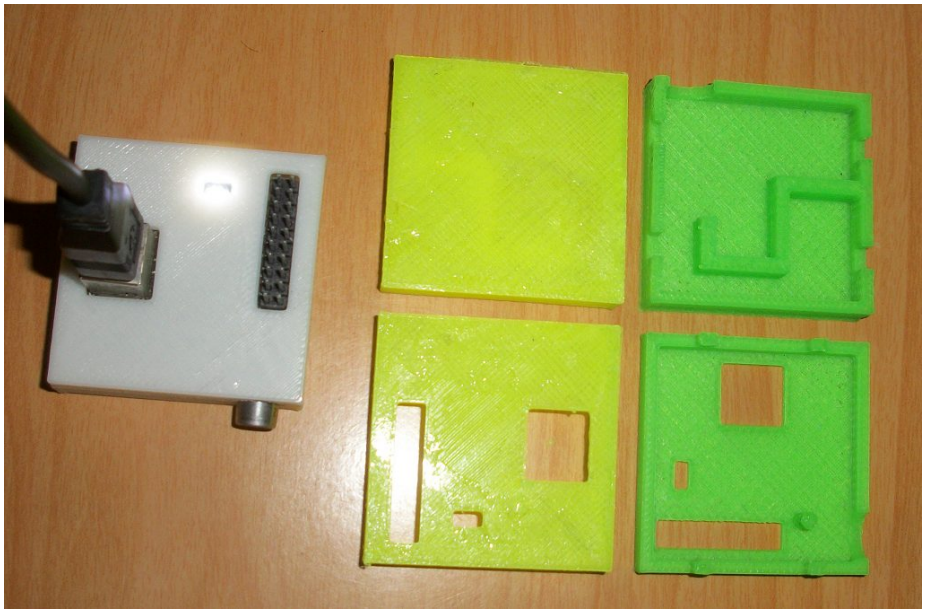


Figure 2.11: Housings designed for Elda noses. See appendix 2.

2.3 Data acquisition

The main function for the artificial noses in Elda biodiesel plant is to record data of the air to store and then process it. Each sensor generates every fixed time a number between 0 and 1023 (2^{10} values or ten bits). The data capture intervals can vary depending on the needs, because the nose reads the sensor output value only when is asked to. A maximum of ten readings per second can be made without risking the nose micro-controller to collapse and not being able to give the response in time, generating in consequence, a delay. The interval time has been fixed to capture data every one second approximately. So, taking in account that fifteen sensors are present in the context, fifteen values are generated by the system every second. Also another time series is necessary, the time stamp when the data has been captured.

A central computer is installed in the Control Room just behind the nose placed there. This computer is in charge of asking the noses for the data every second. Also this computer provides power to the noses to work. It has a windows 7 operative system installed and the software created specifically for the noses, called *OlusElda*. Also a remote control system is installed for easing the dump of the data to the computer where processing is done, which is located in Madrid (GNB laboratory in the EPS in the UAM), as well as to solve the problems that arise, to update the software and also to monitor whether the system is working properly or to restart it if necessary. This program requires a broadband connection twenty four hours a day.

Artificial noses are connected in series to the computer through a USB to Serial Converter Cable. Three different cables are used to connect the noses due to different issues that arose. The signal intensity decreases in function of distance of the nose to the central computer and also is affected by the number of connected noses at the cable segment that take the necessary energy to work. As being a serial cable, the same cable has to pass through every nose that is intended to be connected to it, which makes the cable to be more than one hundred meters long if we want to connect it to all six noses, something inviable. Also the cables have to be wired without interfering in the work of the factory, so when less meters installed will be better. So the path of the cable to the different noses has to be short and with not many noses connected to it. At last, three cables connect the noses to different USB ports in the main computer. Each connection is named as COM3 COM4 and COM5 because of being the names the computer has given them. In the beginning COM4 and COM5 were the only cables installed. after some months a new artificial nose was introduced into the system, so connection COM3 was made. The connections are as follows:

- COM5: In this connection, both load room and centrifuge area noses are connected. For making this connection, the cable has been split into two of the same features, causing it to lose signal strength but still this loss is not as high to make any data to stray.

- COM4: Reactor area artificial noses and the nose inside the control room are connected to COM4. Both noses in reactor area are near one each other, so being connected to the same cable is logical. Control room nose is also installed in this connection because the cable is not so long and there is enough power for the three noses.

- COM3: This connection connects only the nose outside the control area. The reason of only having one nose is that it was the last to be installed so instead of modifying the other connections, a new one has been made.

2.3.1 Data acquisition program

To achieve the communications between the computer and the noses, a specific software is needed. This program is called *OlusElda* and serves as a bridge between the noses and the Computer, automating the data capture process. The first version of this program has been granted by David Yañez.

OlusElda sends a packet through the cable that reaches all the noses connected to it. In this packet is defined the receptor of the message and the data to be returned. When a nose receives a packet it checks if the message is for it or for other nose, if it is for other nose, it does discard it, but when is for the nose receiving the packet, it checks the sensor data asked and sends the value back to the central computer.

For isolating possible errors, the program has been modified making one instance for each connection, so three programs are running at the same time in the main computer, as seen in Figure 2.12. If a connection fails, the program has to be reset without affecting the other connections.

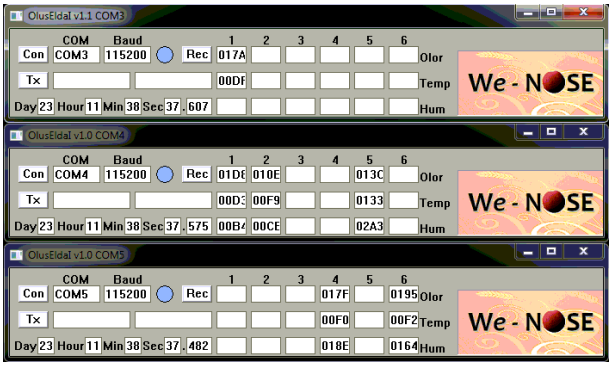


Figure 2.12: *OlusElda* three programs UI working at central computer.

As seen in the Figure 2.13, different buttons and data is presented to simplify and monitor the communication with the noses. There are three buttons:

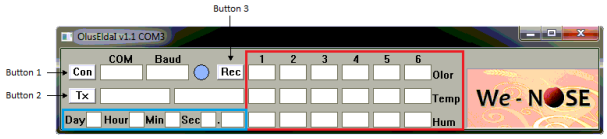


Figure 2.13: *OlusElda* program diagram.

- Button 1: This is the connection button. It sets up the connection defined in the *COM* box, which can be COM3, COM4 or COM5. Other values would give an error. The connection is established through an USB port

of the computer. The baud rate of this connection can be defined in the *baud* box. The default value is 115200 and should not be changed. When there is no open connection, the circle on the right of the button is dark blue, and when the connection is established, the circle turns into a light blue.

- Button 2: This is the transmission button. When the connection is established, this gives the possibility to send an individual packet to the connection cable. There are two boxes on the right, filling the first on defines the nose to send the message and the sensor of which we want the value and the second one displays the value returned. When pushing the button, the message is sent and the answer, if received, is shown.

- Button 3: This is the record button. When the connection is established and button is clicked, the program starts to record the values of the sensors of every nose that is linked to that connection. A new file is generated and the values are recorded as they arrive to the program. When recording, in the red square, the different sensor values are shown in real time, being the columns the nose number and the rows the type of sensor. As no nose has both an humidity and radiation sensor, the third row shows humidity or radiation sensor depending on which is installed in the nose. If no third sensor is installed, the third row of that nose will be blank. The blue box indicates the time stamp of the last recorded values, including day, hour, minute, second and millisecond.

Different modifications were made in the program *OlusElda* to make it fit to the needs in Elda biodiesel factory. Appendix 1 gives an in depth description of those changes in the program.

2.3.2 Output file

The program gives an output of the data registered in a determined format. The name of the file depends on the COM that registered the values and on the time that the recording has started. The file name is as follows: *COM[N] [WeekDay] [Month] [Day] [Hour]_[Minute]_[Second] [Year].xls* where each value in square brackets must be replaced with the following values to have the output file name:

- N: In combination with *COM* gives the source from where the data has been taken. The possible options are 3, 4 or 5.

- WeekDay: Defines the day of the week when the first time stamp was taken. The name is reduced to three letters giving the next options: *Mon*, *Tue*, *Wed*, *Thu*, *Fri*, *Sat* or *Sun*.

- Month: The value is the first three letters of the month when the first time stamp was recorded. The possibilities are *Jan*, *Feb*, *Mar*, *Apr*, *May*, *Jun*, *Jul*, *Aug*, *Sep*, *Oct*, *Nov* or *Dec*.

- Day: This is the day in number when the file was created, having always two digits, so numbers 1 to 9 would have a 0 before. Values can go from 01 to 31

- Hour: Two digits represent the hour when recorded the first value. Again numbers 0 to 9 would have a 0 before. Values go, then, from 00 to 23.
- Minute: Other two digits represent the minute when first recorded, making the values possible from 00 to 59.
- Second: This represents the second when the first time stamp of the file has been taken, as in minute, the possible values go from 00 to 59.
- Year: Four digits represent the year when the first timestamp was taken, such as 2014 or 1990.

An example of file name could be

COM3 Mon Jul 28 13_44_44 2014.xls

Each file registers 84600 timestamps with the values of the sensors, so a file is generated each day. When 84600 time stamps have been registered, the file is closed. When new data is captured, another file is created with new values in brackets depending on the time stamp of this new data. So one file is created every day for each communication port or COM, that is three files a day, each one with its own characteristic name that also identifies the port and the time when was created.

Inside the file, each data capture creates a new line and each value given by each sensor is divided with a tabulator. First, all the sensor values are saved, and a last column with the time stamp is given. The format of this last column is *Y[Year]M[Month]D[Day]H[Hour]:[Minute]:[Second].[Millisecond]* having [Year], [Month], [Day], [Hour], [Minute] and [Second] the same pattern and possible values as in the file name. The [Millisecond] value is composed of three digits representing the millisecond when the data was taken, having the possible values going from 000 to 999. An example of a row in a file is

544 214 179 420 249 206 433 307 630 Y2014M08D04H11:44:47.190

Depending on the COM, different sensors are registered, so there are different number of rows in each file depending on if it starts with COM1, COM2 or COM3. A scheme of the different columns is given below.

- COM3: [Nose 4 gas sensor value] [Nose 4 temperature value] [time stamp] Example:

  541 223 Y2014M08D04H11:44:58.375

- COM4: [Nose 1 gas sensor value] [Nose 1 temperature value] [Nose 1 void value] [Nose 2 gas sensor value] [Nose 2 temperature value] [Nose 2 void value] [Nose 3 gas sensor value] [Nose 3 temperature value] [Nose 3 radiation value] [time stamp] Example:

  505 214 177 342 248 204 595 304 627 Y2014M07D29H11:45:11.701

- COM5: [Nose 5 gas sensor value] [Nose 5 temperature value] [Nose 5 humidity value] [Nose 6 gas sensor value] [Nose 6 temperature value] [Nose 6 humidity value] [time stamp] Example:

  402 238 468 507 236 434 Y2014M05D06H11:45:06.127

The void values indicate that those values are not linked to any sensor, but they are saved to not change the program when a new sensor is introduced in the nose. Those columns can be erased because they don't give relevant data to the system.

2.3.3 Event registration

Four kinds of events are registered: methanol load, caustic soda load, reactor startup and reactor unload. All these events are registered manually by the workers in the plant. So each time methanol or caustic soda is loaded or the reactors are started up or unloaded, the workers register the date and time when the event has occurred. Each event has its own characteristics and is important to understand how they affect the factory environment:

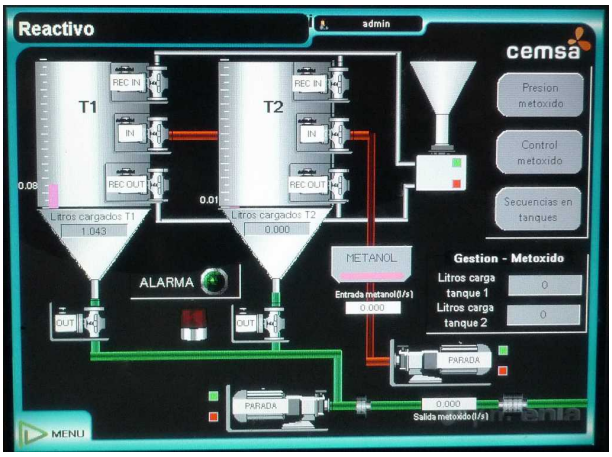


Figure 2.14: Load room control panel (Image granted by F.B Rodriguez).

- Methanol load: This event happens in the load room. Methanol is loaded in the system through a pipe, so the noses should not identify the event properly. In case methanol reaches the air, the environment becomes hazardous, this is a leak. The noses then should not detect the methanol in normal conditions, so, when detecting it, we have a leak. In this situation, a fast response time is desirable and the first nose to detect it should be the one located in the load room. These events are registered in papers which register two time stamps, one when the event starts and another when it ends. In figure 2.14 the load system can be seen and is observed that the methanol never reaches the air in normal conditions.

- Caustic soda load: As methanol load, this event happens also in the load room. Again, it is registered in paper manually and registers the time when the event starts and when it ends. Soda is introduced in the load room manually through a funnel as can be seen in the Figure 2.15. So with this system the soda reaches the air and the noses should react to this event.

- Reactor startup and unload: This process is automatic because of being complex and including different phases. The basics of the reactors are to mix methanol and caustic soda to produce methoxide and combine it with the oils. Then a reaction process that takes hours starts. These two events mark the beginning and the end of reactor activity. In Figure 2.16 a general picture of the reactors can be seen. The process is computer

Figure 2.15: Funnel used to introduce caustic soda (Image granted by F.B. Rodriguez).

controlled but the start of the event is defined by the workers, so again the registration of this event is manually registered.

Figure 2.16: General picture of reactors (Image granted by F.B. Rodriguez).

For all the labels in the registration papers, a column named comments allow the workers to inform of relevant extra data. As the registration is manual, the event start can defer when is detected by noses and when is registered in the papers because of the times of the noses and the workers not being synchronized or because of them not putting the exact time.

2.3.4 Recording periods

Since the noses were installed there has not been a continuous recording. Each continuous recording is called recording period or period. Different periods arise because of some reasons, some are environmental issues or program fails and updates and other are because of changing the topography of the noses.

There have been different periods of data recording in Elda plant related to topography changes, linked to the three visits that were made to the factory. In each visit new noses were installed in the system.

- First visit: In this visit, the first nose located in the load room is installed. Only a temperature sensor is attached to this nose and the *OlusElda* program with the computer are installed in the control room. Only a COM is created with a single program executing. In this visit, caustic soda and methanol load timestamps begin to register. After six months of continuously capturing data except for some outages solved by rebooting the system, the nose collapses. This collapse happens because the nose has a fan to control the air flow to use it as another variable, and that causes the inlets of the sensor to block with caustic soda. This makes and another visit to the factory needed.

- Second visit: Four new noses have been installed and radiation and humidity sensors are activated. Two noses are installed in the reactor area, other in the control room and a last one in the centrifuge area. The nose in the load room is replaced, realizing that the sensor is blocked with caustic dust suspended in the air. Different nose sensors are in different noses to allow the network to detect a wider range of odors. The program is modified making two versions of it, one for the COM5 and other for the COM4. The fans are removed from the noses to prevent the collapsing of the sensors. The noses work correctly for six months without problems except for outages that are solved remotely. Having different ports with different time captures, some modifications in the program were needed, such as changing the name of the output file and adding a new column with the time of each register and adding milliseconds to the time stamp.

- Third visit: In this last visit, the nose outside the control room is installed and also the housing for all noses (see Figure 2.11). A new modification in the program is created as also a new instance for the new communication port COM3, not making any substantial change on it. Reactor startup and unload event times are started to be registered.

The second period will not be used in the experiments because it does not provide extra information than the third period. Also during that period the plant had very low activity and there were many power outages.

2.3.5 Data processing

For classifying the time series generated in the Elda biodiesel plant, Ramón Huerta granted different classification software. These programs consisted in different modifications of support vector machines (SVM), specifically a dynamical multidimensional time series classifier based in support vector machines (DSVM) [Huerta et al., 2012], an autoregressive support vector machine as kernels of a SVM (ARSVM) [Vembu et al., 2012], a generic SVM instance (SVM) and a SVM with cross validation (CSVM):

- SVM: A generic support vector machine classifier has been granted by Ramón Huerta. A support vector machine constructs a hyperplane or set

of hyperplanes in a high-dimensional space that divide the sample space in different subspaces. The hyperplane has the largest distance to the nearest training data point of any class.

Whereas the original problem is placed in a finite dimensional space, sometimes the sets to discriminate are not linearly separable. For this reason, the original space is mapped into a higher-dimensional space, making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM are designed in terms of a kernel function k(x,y) to make easier the operations from the finite space.

- CSVM: Applying cross validation to the SVM we can improve the classifier. This modification of the original classifier consists in cross validating the data. For that, training and test inputs are put together and different trainings and tests are generated following the next scheme, the data is divided into x groups of the same number of time series and then SVM runs x times having each time a different group as test set and the rest of groups as training. The best execution results in the classifier to chosen.

- DSVM [Vembu et al., 2012]: This classifier combines nonlinear dynamical systems with the classical support vector machines, performing the capabilities of both methods used independently.

  A nonlinear dynamical system is defined as a set of nonlinear differential equations. For the dinamical support vector machine, Rössler attractor was chosen [Rössler, 1976]. The Rössler attractor has the advantage of being dissipative, which means that the volume in the phase space is not conserved and is usually contracted, forming complex trajectories. Even being a deterministic system with low dimensionality, the Rössler attractor is unpredictable, making it a good nonlinear dynamical system. The Rössler attractor is modified to become a non autonomous system, depending on the input time series signals, so we can see the system as a feature extractor for the times series. Different dimensions of the times series are related each other by oscillators [Steven H., S., 2001] using the nearest neighbors concept, so each dimension is related with its previous and next dimension.

  When trying to solve the system, each pair (time series and label) used for the training generates a set of solutions. Now, the SVM concept can be applied to those sets of solutions, creating separating trajectories delimited by a function dependent on the input and control parameters of the nonlinear system similar to the separating hyperplane in the SVM classifier.

- ARSVM [Vembu et al., 2012]: For each time series an autoregressive model is made [Cuturi and Doucet, 2011]. Evaluating errors on pairs of autoregressive models gives a high or low value for each model depending on the capabilities of each model in predicting both time series. If both are well predicted, then kernel evaluation has a high value, but if one or both time series are bad predicted, low values are given. This way is not necessary to calculate all the parameters spectrum, but only for each different model independently.

2.3.6 Data formatting

The .xml files that are generated by the *OlusElda* program don't have the proper format for working with the programs granted with Ramón Huerta, which require an special structure. For formatting the time series, two different programs have been created with different capabilities:

- Unify files program: This program grabs the original .xml files and concatenate them in an intelligent way, predicting the names of the next files automatically and also having the capability of unifying different COMs in only one file normalizing the timestamps. As the different *OlusElda* program executions can register different number of time stamps, the normalization to unify the different COMs can be done in two different ways: one is reducing the points to the COM with least points recorded and other is normalizing data to one point per second in a linear way. In both approaches, the output for this program is a single file with all the values registered by the noses without having any blank value, but the first option is preferable to avoid the apparition of artifacts. The need for applying this program to a set of .xml files is that they have to belong to a recording session without breaks. Different session data can be extracted independently but sharing the same format and unifying it afterwards for different data processing.

- DSVM formatting program: This program grabs a file of timestamps, as Unify files program returns, and creates a file with the correct format for being used with the classifiers, including assigning labels to each time series generated, being able to only generate two labels, 1 or 0 per execution. The input file has to have as many rows as wished representing each one a time stamp but with one condition, the time between consecutive timestamps has to be the same in all rows. The columns represent each sensor recording and can be as many as desired, but no blank space is allowed. An option file is necessary for executing the program, which consists in two parts, the first part is a number which defines the length of the output time series (indicates the number of rows that conform a time series regardless of the time between records) and the second part that is a list of numbers that represent the row where each label classified as 1 starts. The time series classified as 1 are generated from each start point defined in the option file with the length also defined, while the time series with label 0 are generated using the rows not used for creating the time series labeled as 1 with three conditions: they have to be as long as defined in the option file, only one row can be used once for a time series independently of their label and the rows used have to be consecutive for each time series generated.

Both are console programs to be easily included in scripts to automatize all the data processing. For making all the formatting process, different linux scripts were made. For classifier execution, linux scripting was used to automatize all the process. Also linux and matlab scripts were used to generate the graphs used in the next chapters.

Chapter 3

Results

3.1 Specific event and general activity monitoring

After installing the nose sensor network in Elda's biodiesel plant and executing the program to acquire the data, it was necessary to check if the sensors received representative data. At glance the data was generated and the three instances of the *OlusElda* program were executed correctly.

The first test to check the activity in Elda plant and seek the capabilities to detect security issues reviewing if the data is sound is to make perform a visual inspection of the main features of the time series. This section will focus on highlighting the features of relevant events to check if some of these characteristic patterns can be defined before going to automatic detection and analysis processes.

As explained before, three recording periods are defined based on the number of noses present in the plant. Only the first and the last ones will be used due to being the interim period too short in comparison with the other two and not having as much activity as the other ones. In the first period only one nose is installed inside the load room with chemoresistive and temperature sensors while in third period all the noses are present and recording.

3.1.1 Leakage events

The main purpose for installing artificial noses in Elda biodiesel factory was to detect leakages in the load room, that is why the first nose was installed there. A leak consists in methanol reaching the air in the load room without any control so we can expect the nose to react saturating and increasing its output value. The ideal method for detecting leakages would be to force different leaks with controlled environmental characteristics, but in a completely functional factory such as the one where the noses are installed it was impossible to force a leak because of the risk involved and also because it would lose activity time in the plant, something not viable for the business. So, for having time series from leaks for preparing the training set, the only option was to wait until some non intentional leaks appear. Leaks are not common events, and only one was successfully registered in the first period. Only having registered one leak event makes very hard the detection of them in the future, so the focus of this master thesis has come into other areas until more leaks appear, something that has not come to pass.

Figure 3.1 shows the time series that represents a leak in the Elda plant. It is easy to see in the graph that the sensor reacts to the event increasing its output value. Being methanol the gas that reaches the air, the reaction of the nose is sound.

The leakage is unpredictable, so is detected after it happens. In the plant is impossible to know when exactly the event occur, so the time stamp of the beginning is not accurate. As seen in Figure 3.2a, the nose reacts in a chaotic way having much noise represented as very sharp steeply ups and downs even before the label that marks the beginning of the event. Anyway, the increase that saturates the nose happens afterwards. More examples of leakages could help to see the exact reaction of the nose when exposed to them.

Other important point to take in account is when the leak finishes. Different

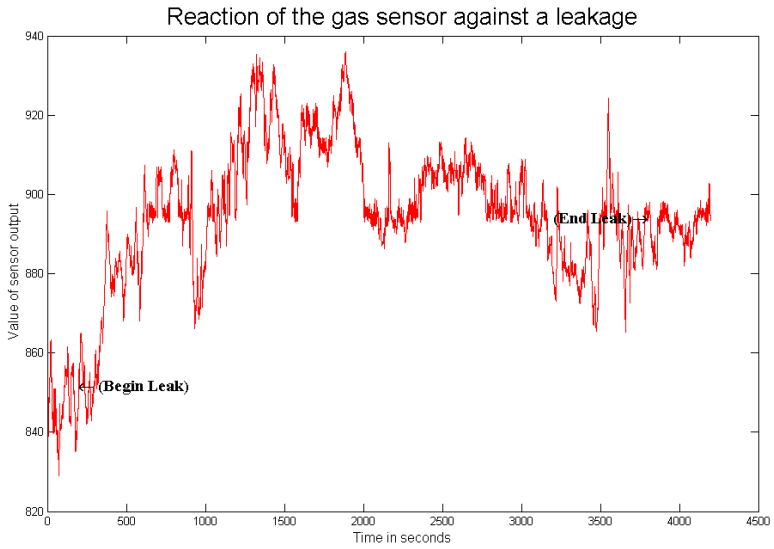
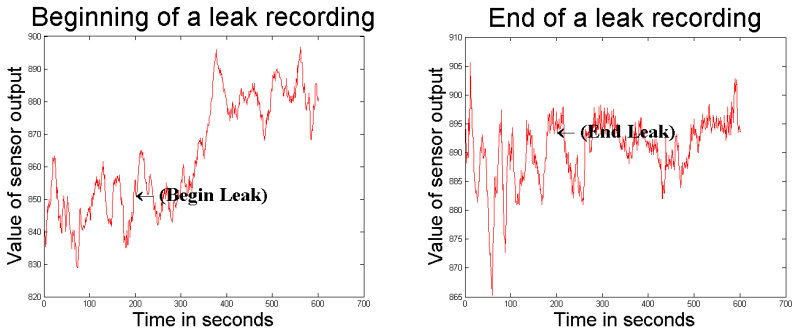


Figure 3.1: Graph of the time series during a leak event. For all graphs in this chapter, sensor values are expressed in arbitrary units.



(a) Beginning of the leak.

(b) End of the leak.

Figure 3.2: Blow up of the time series beginning and end of the leak event.

possibilities arise: one is that the leak finishes when it stops to eject harmful gas to the environment, other one is when the gas concentration in air reaches normal values. What is certain is that the end of the leak can not be defined as precisely as the start. A problem when a chemosensor saturates is that when the exposure to the gas that saturated it ends, the recovery time is very high, in the order of minutes. Taking in account that, in a leak, the gas that saturated the nose does not disappear suddenly, the recovery of the nose is even slower. So as can be seen in the Figure 3.2b, the end of the leak is not clear, and also in Figure 3.1 the recovery of the nose can be seen as a slow process.

The graphic representation of the time series of the leak can give us some

conclusions: A leak is an event that can be detected by simple sight in the time series. Also is a long event of around one hour or more. The beginning and the end of the leakages are not certain, but the end of a leak is not as important as the beginning is. After a leak, the nose will be saturated for a long time being hard to detect anything afterwards, because the nose will give similar responses to different events.

3.1.2 Methanol and caustic soda loading events

As being the main purpose of the artificial nose sensor network and monitoring Elda plant to detect leakages and not having enough leak events, it is proposed to monitor other kind of events. Leak events occur in the loading room detecting events in this room can give us an approach of the system capabilities for leak detection. Also when loading methanol and caustic soda, leakage risk raises because there can be a failure in these loading processes generating a hazardous situation. Then, detecting load events can bring us closer to leak detection.

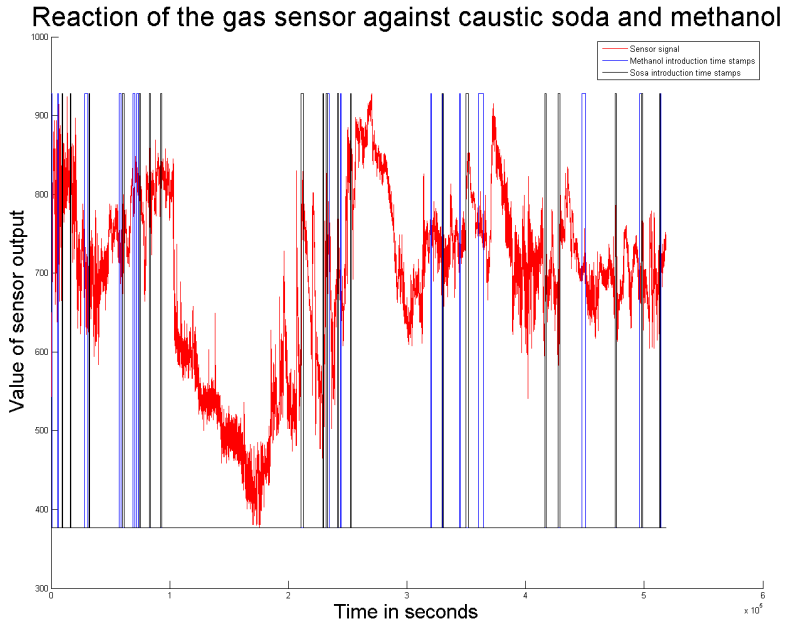


Figure 3.3: Time series with the beginning and end of the soda and methanol load events plotted. Periods with no events show a decrease in the nose signal.

A factor that affect the artificial noses sensors is the caustic soda and methanol load. A general view of the caustic soda and methanol events can be seen in Figure 3.3. As we can see there is no logical order in the load of soda or methanol, so we can assume that the events are independent. Also the load of soda an methanol does not go in pairs, sometimes two and even three consecutive caustic soda loads take place. Other interesting point is that caustic soda load events are more typical than methanol ones.

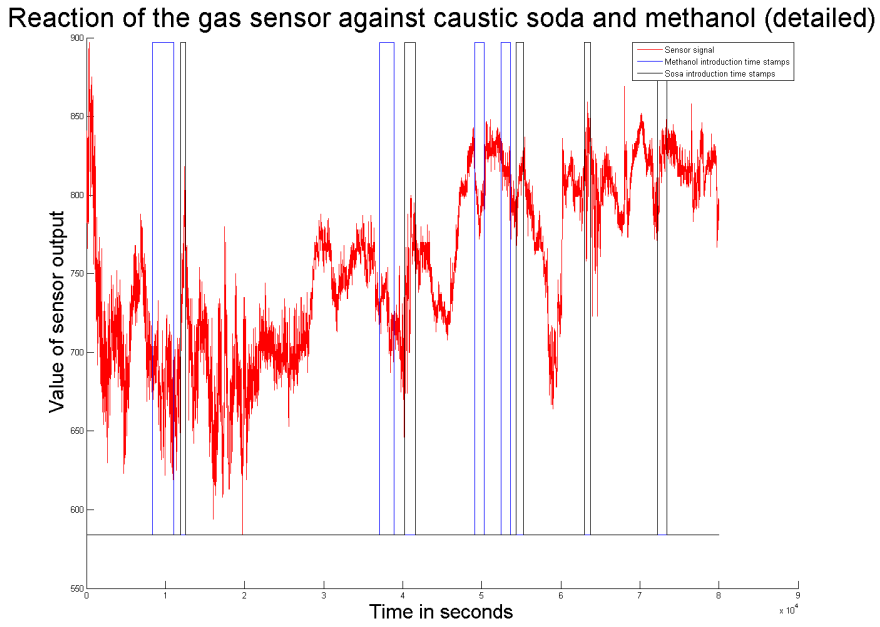


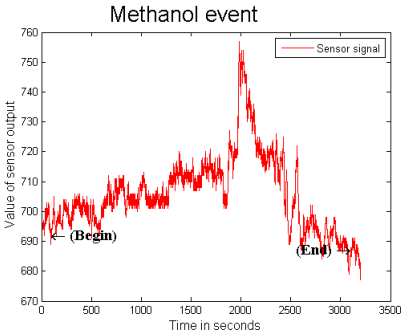
Figure 3.4: Blow up of graph in Figure 3.3. Event lengths are variable and there is no clear appearance order.

In Figure 3.3 we can observe that there are periods where events accumulate. We can understand that these periods are activity ones or when the plant is making biodiesel. In the other periods no loading events occur. Watching the graph we can see that, during these break periods, the nose lower its sensor value, what means that there are lees components affecting the sensor in the air. So is logical to think that on those periods there is no activity and the plants air is cleaned gradually. These defined periods are not of fixed length as is easily observed in Figure 3.3.

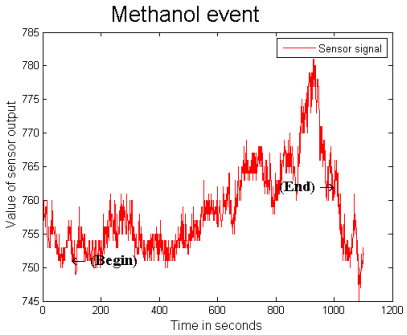
A zoom in the graph can be seen in Figure 3.4 and here we can easily check that soda an methanol load events have apparently no order of appearance and that caustic soda an methanol load events are not interleaved, but more than two same class events can take place consecutively. Also in Figure 3.4 is observable that events do not have a fixed length but this varies, being methanol load events longer than caustic soda ones.

We now will discuss the methanol load events and how they affect the time series. We will use period one temporal series to discuss it where only one nose with one chemosensor and temperature sensor is placed in the load room. In the period one, as only having one nose and being this one near the place where the methanol is loaded, the graphs can give a good overview of the effects of methanol load in the sensor.

In Figures 3.5, 3.6 and 3.7 we observe six different time series related to six methanol load events. The time lapse between different events is less than a week, so the drifting due to aging of the sensor is not affecting the values.

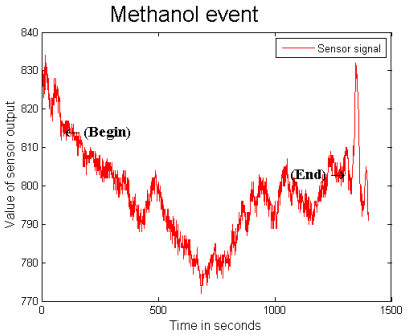


(a) Methanol load graph 1.

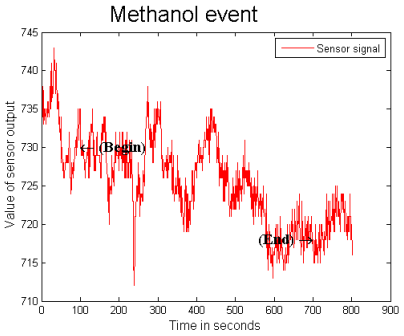


(b) Methanol load graph 2.

Figure 3.5: A pair of examples of methanol event time series with a clear peak during the loading event.



(a) Methanol load graph 3.



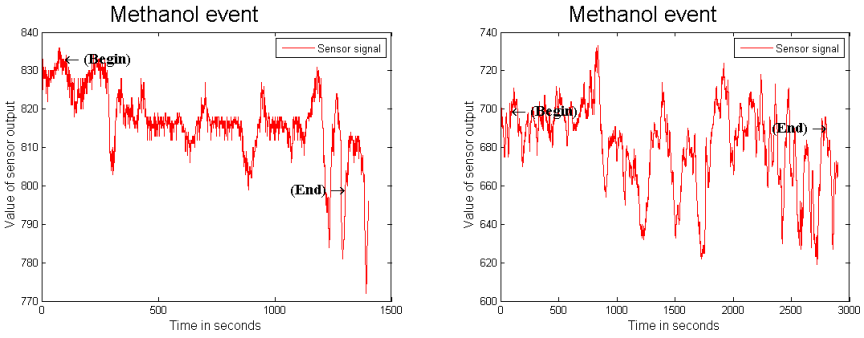
(b) Methanol load graph 4.

Figure 3.6: Two examples of methanol event signals characterized by a decrease in the sensor value.

As we can see, there is no detectable pattern at a glance. The first two events (Figures 3.5a and 3.5b) increase the values at the beginning of the load, which ends in a high peak, but this does not happen in the other four graphs. This could happen because of the nose being saturated, but the graph in Figure 3.7b starts at a value 700 while the graph in Figure 3.5b starts the event at a value of 750, so the option of being saturated is discarded.

There is neither an increase nor decrease forced by the event in the value of the sensor since the event starts until it ends. As we can see, in Figure 3.6b the value at the end of the event is lower than at the beginning of it, in Figure 3.7b the values are similar at the beginning and at the end of the event and in Figure 3.5b the value is higher when the load stops than when it starts.

As mentioned before, there are no characteristic peaks or valleys shared by most events. In Figures 3.6b, 3.7a and 3.7b the time series look more unstable than in Figures 3.5a, 3.5b and 3.6a, so when loading methanol to system is



(a) Methanol load graph 5.

(b) Methanol load graph 6.

Figure 3.7: Methanol examples with not clear decrease or increase in the signal during the event.

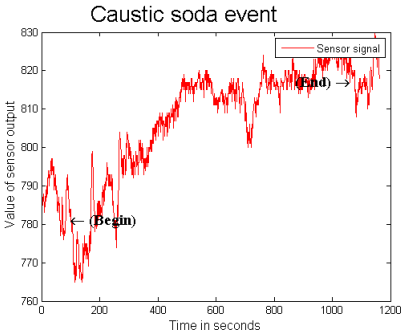
not a characteristic response to have noise in the sensor values. This makes a template based classifier to not be able to classify this event. Also it seems that the nose response when exposed to an event depends on the previous history of the signal, so a classifier has to take in account this feature. Also not in all the loads the same methanol quantity is inserted, so it would be interesting to insert this variable to the classifier.

As was done with methanol load, we will discuss caustic soda load events and how it affects the temporal series. Again, period one time series are discussed to check for different characteristics.

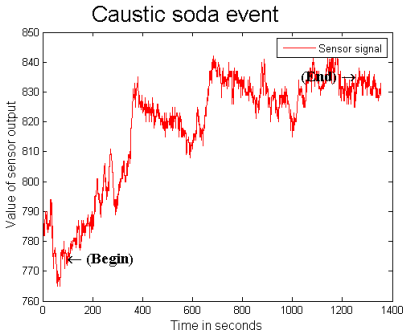
In Figure 3.8 we can see two graphs that represent the generic response of the sensor to soda load. These graphs are extracted from close in time events to not have significant differences due to drifting. The first installed nose stopped working correctly because the sensor was blocked with caustic soda which was agglomerated in the inlets. Is important to take in account that noses placed in the load room, where caustic soda is loaded, suffer high drifting over time, possibly changing the reaction patterns to odors of these sensors.

Methanol load events have a wide variety in sensor responses. But, in caustic soda load events, even being each one completely different from the rest, the sensor reaction is very specific. This reaction consists in a visible rise in the sensor response. It begins with a steep climb that gradually decreases as the nose approaches to saturation values. In Figure 3.8a it is visible a general characteristic curve that represents the soda load event in the time series registered by the noses. In very few cases the curve is so clean and usually different environmental elements, previous values of the sensors, quantity of soda introduced, and not controlled events interfere in the graph as can be seen in Figure 3.8b. Some of these artifacts can not be controlled as being an open world system in an uncontrolled environment while others can be introduced in the classifier to make it take in account this data. But anyways we can notice that the nose is reacting to something, in this case the caustic soda load event.

Also is easily seen that in these events, the start of them is not well determined because of different issues, such as not being synchronized the timestamps of the sensor caption and the events or not having a in-depth control over the



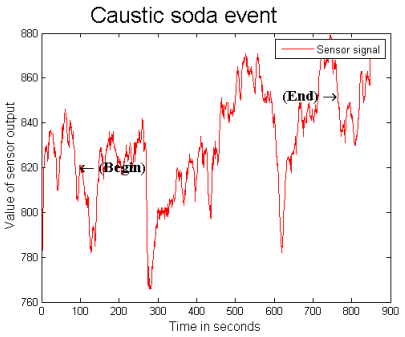
(a) Soda load graph 1.



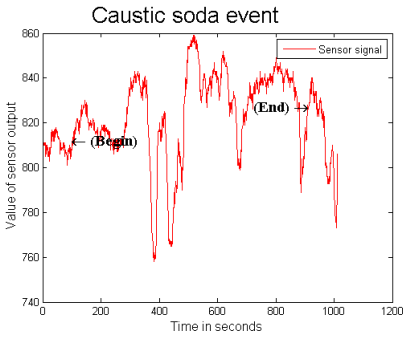
(b) Soda load graph 2.

Figure 3.8: Two typical soda event examples. The value increases during the events.

method of registering the event timestamps. As being events of ten to twenty minutes, not having synchronized the timestamps gives a minimal error of a pair of seconds at most. But the problem comes when factory workers register the data, because usually the same person doing the load is who makes the record, and hence a number of human errors can arise as to round time, register it wrong because of registering the time after loading , or accidentally forgetting to register it and then set an approximate time, this last one being unlikely but possible.



(a) Soda load graph 3.



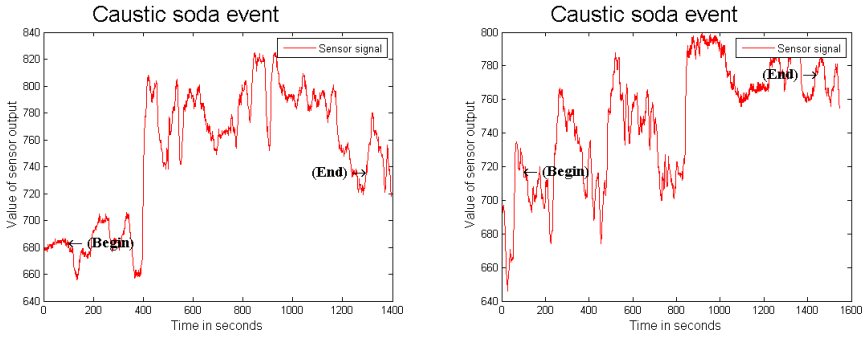
(b) Soda load graph 4.

Figure 3.9: Different soda event sensor response examples with saturated noses. The values do not increase too much during the event.

As mentioned before, saturation affects the nose to not rise more its values. This saturation is not a clean cut, but it happens gradually, making, in some values, harder for the sensor to go on rising the output, hence making the response less visible than expected. As being gradual, we cannot define a value when the nose can be considered saturated. We can define the nose to have a saturated reaction when values are over 800, but other factors can make the

nose enter this state over or below this value.

As can be seen in Figure 3.9b, the sensor value does not rise too much during the event showing a small difference between the beginning and the end values. There are artifacts that make the nose dramatically decrease its value and back up, sometimes suddenly and other gradually. Probably this is a side effect of a saturated nose making easier for the sensor to decrease values than increasing them, but there is no proof of this hypothesis. What is certain is that the nose keeps generally rising even when being saturated, but we must take in account that the reaction to soda events are dependent on the value at the beginning of the event as part of the previous sensor history.



(a) Soda load graph 5.

(b) Soda load graph 6.

Figure 3.10: Two examples of atypical behavior in soda event time series. High increase and low decrease gradients are present in both graphs.

As being an uncontrolled environment, not all the events are so clear in the output of the sensors to caustic soda loads. We have seen it when the nose is saturated, but sometimes even with the nose not being in this state, the response is not so clear. In Figure 3.10a can be seen that the nose gives a plain response at the beginning of the event and suddenly the value rises dramatically and afterwards reacts as being saturated. In Figure 3.10b the nose acts as being saturated but with values that are not high enough to determine the nose in this state.

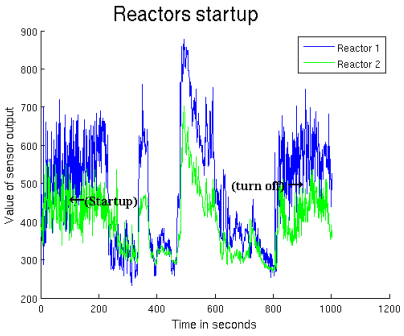
As we can see in the six time series, the caustic soda event is very characteristic, even with atypical behaviors on the nose response due to different uncontrolled environmental events or previous history values sensitivity. The nose always reacts rising its value even when different artifacts affect the environment. So is easily seen at glance the soda events, the response of the electronic nose is sound for caustic soda load events. As seen in methanol load events, in soda ones we again have the answer determined by the previous history of the signal, and also different caustic soda quantities are introduced in each event. Because of these defining characteristics of the event, is not possible to detect caustic soda loading events with a template based classifier, but with a classifier that takes in account soda quantity load and previous history of the sensors.

3.1.3 Reactor startup and unload events

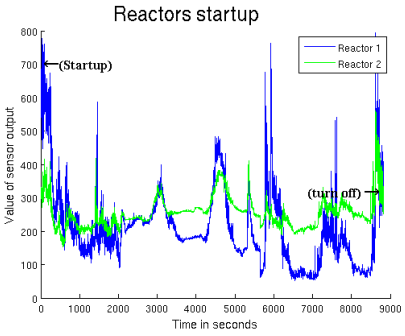
Events of startup and unload of the reactors are registered late in the third period, so we can only introduce graphs of that season. In this period the six noses are present, placed through the whole plant, recording sensor sensibility, but the reactors noses are the ones that are focused in detecting these events, so for showing clear graphs, only these two noses signals will be shown to give a global approach of the effect of the reactors startup and unload in the sensors response.

Reactors are being used continuously for long periods of days and even weeks. During these seasons it may take some hours between one use of the reactors and the next one, so usually those periods are related with a high activity in the response of the sensors. When there is no stored used oil, the factory is not working, and those periods can also be of weeks, but usually they are defined by a length of days.

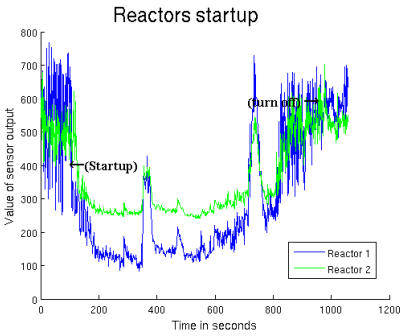
Startup events are defined when the reactors are being loaded and started, so in these events is also included the loading of the mixture.



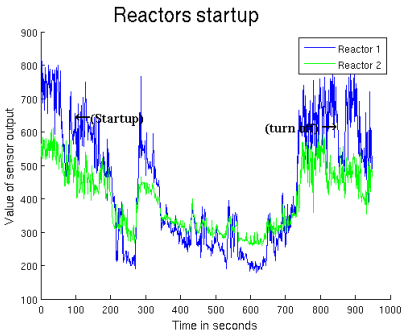
(a) Reactor startup graph 1.



(b) Reactor startup graph 2.



(c) Reactor startup graph 3.



(d) Reactor startup graph 4.

Figure 3.11: Two examples of reactor startup event time series. A decrease in the signal can be observed during the event.

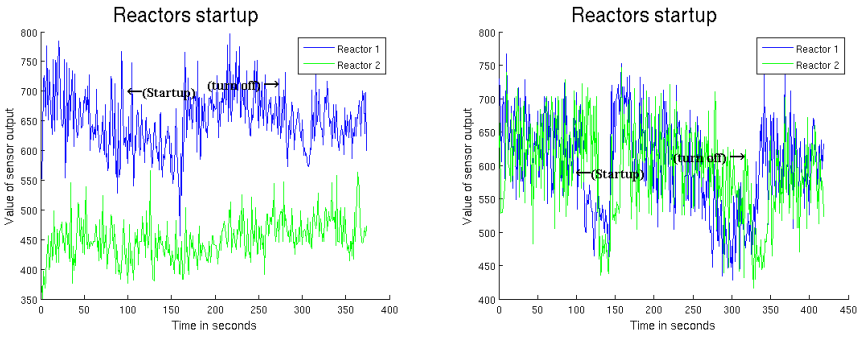
In Figure 3.11 we can see four graphs representing the time series of the two reactor noses during four different reactor startup events. As we can see,

shortly after the event has started, the sensors signals relax demonstrating that during the reactors startup there is not much presence of compounds that react with the noses. During the event and the low activity in the noses, time to time there is a peak that grows fast and goes back to the previous values. Probably these peaks exist because of a punctual event that is happening in the plant. As we don't have a control over the environment, different events can befall without knowing they are happening and even less knowing the reason of them. The recovery slope of these peaks is sometimes fast and other slow, this is more likely to be due to the possible difference in the reasons that cause the peaks.

When the some peaks reach their maximum value, the nose is near saturation or saturated. Probably this makes the recovery of the nose slower, but also means that the unknown source events have a high response in the noses, probably because of its proximity to the artificial noses or maybe because of being events that affect strongly to the air in the environment. On the other hand the lowest values are around 200 or 100, which means that there are almost no compounds reacting with the sensor at those moments, so startup event show responses across almost the whole sensor value spectrum.

The valleys generated during the startup of the reactors can be probably because during the load and startup of the reactors no other tasks are being carried out. Also all the pipes that lead to the reactors and the reactors themselves are isolated to not allow hazardous substances to reach the air and contaminate it.

Both reactor signals suffer the same ups and lows during an event, and practically at the same time, which is sound. This means that both noses react to the same events, as it is desired. But on the other hand, we can notice that Reactor 1 nose have wider value response than Reactor 2 sensor. In Figure 3.11c this is easily observed for the low values, Reactor 2 signal in the valleys is clearly higher than Reactor 1 signal. In Figure 3.11d this effect can be seen for the high values at the end and at the beginning of the graph. The reason may be surely due to the different chemosensors of both noses. While Reactor 1 nose has a model 2611, Reactor 2 nose has a model 2602.



(a) Reactor startup graph 5.

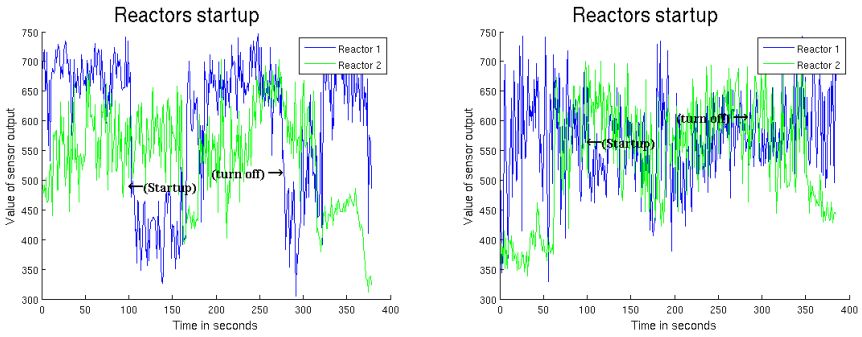
(b) Reactor startup graph 6.

Figure 3.12: Short reactor startup event time series examples. Sensor values do not seem to reflect the event.

The reactor startup events are not all the same length, almost everyone lasts two hours more or less but some are shorter having a length of less than an hour.

In Figure 3.12 a pair of short startup events are shown in two graphs. Their reduced duration can be seen comparing them with graphs in Figure 3.11. As being short events, the valleys don't have time to appear, this is probably because there is not enough time to make the air become clean. The absence of the valley is not due to the nose recovery from a saturated state, because during the event there are many fluctuations and no apparent general decrease in the values. Also there is no manifest difference on the signal fluctuations before, during and after the reactor startup events. This is probably because during the period where the factory is active twenty four hours a day there is no break in the response of the noses.

In Figure 3.12a there is a significant difference on the values of each sensor, being Reactor 2 nose signal lower than Reactor 1 nose output. As mentioned before, this is because of having each artificial nose different sensors they react with different sensibility to gases suspended in the air.



(a) Reactor startup graph 7.

(b) Reactor startup graph 8.

Figure 3.13: Atypical reactor startup event time series examples. Signals in both graphs are not correlated.

Not all the events have a well defined response, some have atypical time series during a startup event. In Figure 3.13 two unusual reactions can be seen.

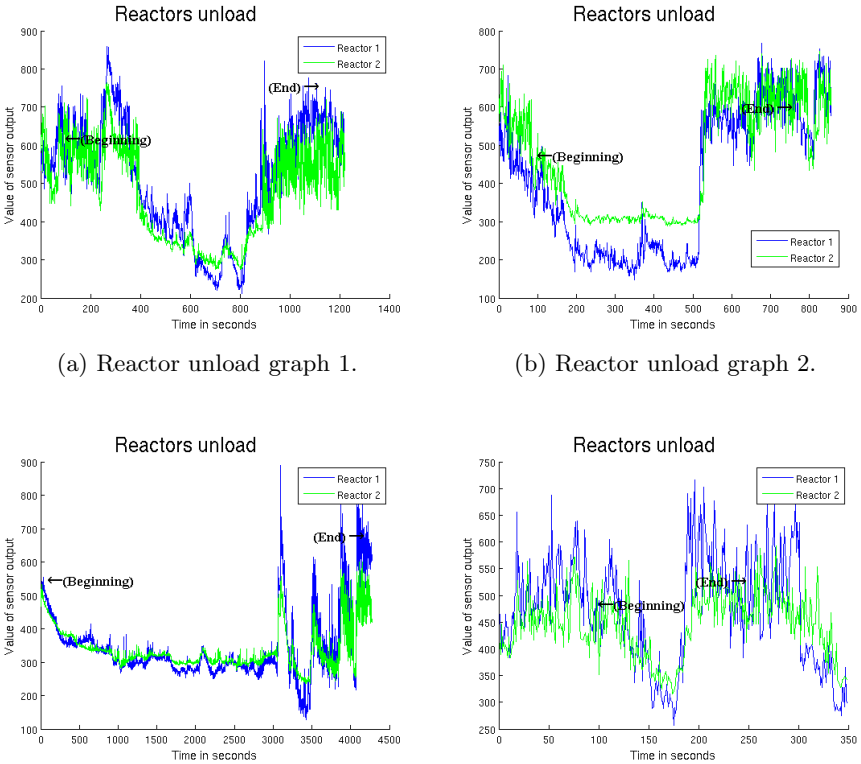
In Figure 3.13a we can observe that reaction on both sensors is not the same. While Reactor 2 nose has a continuous fluctuation, in Reactor 1 time series we can observe interspersed stripes of high and low values. Also at the end of the graph the values of green curve go down abruptly while in the blue time series the values remain high.

In Figure 3.13b, before starting the event and at the end of it, Reactor 1 signal keeps fluctuating while Reactor 2 nose has low values. Also, during the event, the difference between maximums and minimums in blue time series is lower than in the green one.

In the graph of Figure 3.12b, being a short event and therefore having more resolution, a retarded reaction betwixt both noses is seen. The physical distance between both Reactor 1 and Reactor 2 noses is of four meters being both of them surrounded by reactors below them. Each reactor has a determined capacity, and there are sixteen reactors. The plant offers the possibility to only use certain reactors if there is not enough mixture to fulfill all of them. Probably, in Figure

3.12b, the reactors activated for being used are in the part of Reactor 1 nose, having this one faster response to changes in the air than Reactor 2 nose.

After the chemical process made inside the reactors, the biodiesel has to be unloaded and stored in a pair of tanks to allow the reactors to make more fuel. This action is registered as reactor unload. We are now discussing different time series recorded during these reactors unload events.



(a) Reactor unload graph 1.

(b) Reactor unload graph 2.

(c) Reactor unload graph 3.

(d) Reactor unload graph 4.

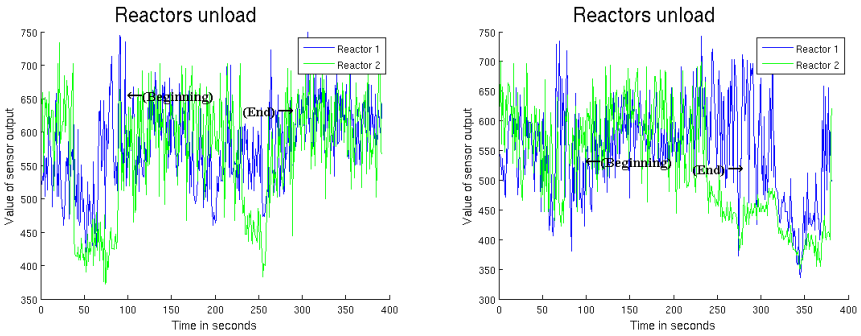
Figure 3.14: Examples of typical reactor unload event time series. A decrease in the signal happens after the event start and also there is high sensor activity near the end.

In Figure 3.14 four graphs show the time series recorded during four typical unload events. As we can see, the response of both noses is similar to when starting up the reactors, as can be compared with graphs in Figure 3.11. In these graphs it can be seen that, a short time after the event has started, the noses relax and have lower values forming a valley that time to time has some peaks. As mentioned before, these peaks are probably made by unregistered events happening in the plant.

An interesting element present in all the time series is that, before the event ends, the noses start to register high activity with large and fast fluctuations, so we can assume this as an intrinsic characteristic of unloading events. Also the variety of values during an event sweep almost the entire spectrum of sensor

values.

In the graph in Figure 3.14a Reactor 2 sensor register lower values during the large fluctuations at the beginning and at the end than Reactor 1. Also in the graph in Figure 3.14b during the characteristic valley of the event, Reactor 2 nose registers higher values than Reactor 1 sensor. So we can conclude that the range of values of Reactor 2 sensor is smaller than the range in Reactor 1 nose. This can be contrasted in the graphs in Figures 3.14c and 3.14d.



(a) Reactor unload graph 5.

(b) Reactor unload graph 6.

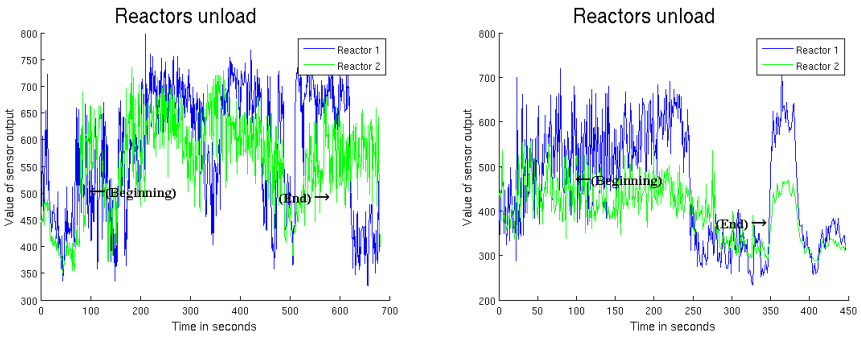
Figure 3.15: Different short Reactor unload event time series examples. The event is not distinguishable because of its duration.

As happened with reactor startup events, the length of unload events is variable, having a media of near two hours but having some short events such as the ones that can be seen in graphs on Figure 3.15. The signal during a short unload event does not decrease to generate a valley. The signal before, during and after short unload events is the same, very fluctuating, so it becomes impossible to distinguish when the event starts or ends. These fluctuations have very wide range with high values at the maximums and low values a the minimums.

In Figure 3.15b it seems that the characteristic valley generated in reactor unload events is generated after the event has ended. Maybe this valley is because of the event or it can be due to other reasons, but it can be possible that the valley is formed a fixed time after the start of the event and this time is greater than the duration of the event, but we can not confirm it.

Also not all events have the same nose reaction, but different non expected responses can arise. In Figure 3.16 we can see two graphs with a pair of examples of these weird reactions to reactor unload events.

In the graph in 3.16a the signal fluctuates during all the event even being this long enough to show the characteristic valley. Probably there is some special activity in the plant. Also here there are short and abrupt valleys in both noses, but it seems that Reactor 2 nose detects the air changes before Reactor 1 nose. In Figure 3.14b the order is the opposite. The most logical explanation is to think that in Figure 3.16a the source of the odor that makes the noses to react is nearer to Reactor 2 nose than to Reactor 1, making the two time series to be delayed. At the end of the same graph, it can be seen that one signal falls while



(a) Reactor unload graph 7.

(b) Reactor unload graph 8.

Figure 3.16: Atypical reactor unload events time series examples. Both sensor responses are not correlated and the characteristic valley for this kind of events is not clear.

the other remains with fluctuations. Again this can be due to events that affect one nose and not the other.

In graph 3.16b we can see that Reactor 2 nose reacts with smaller range of values than Reactor 1 nose. Also when the valley is defined, the blue line descends abruptly while the green one has a lower descent gradient, probably because of being different sensors and being Reactor 1 nose sensor more sensible to air composition changes. Again it can be seen that the peak at the end of the event is more pronounced in a signal than in the other.

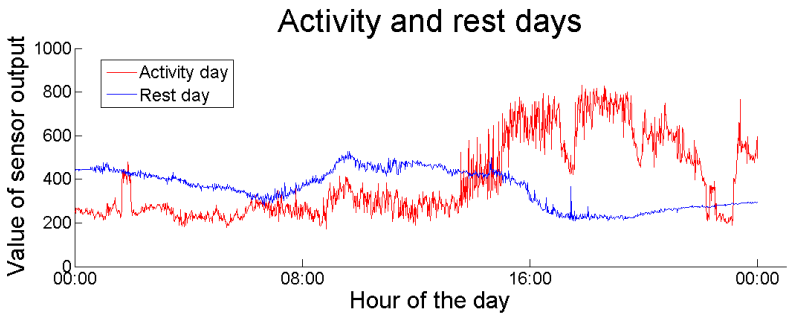
We can see that reactor startup and unload events are not registered so differently. Maybe in future researches we can unify both events as a single type, depending on the capabilities of the classifiers to identify them. These events are also affected by the previous history of the nose and also different quantities are loaded depending on the event, so for all events (caustic soda an methanol load and reactor startup and unload) a classifier that takes in account the previous history of the signals and the quantity defining the event will classify better. For this, a classifier with a learning process can give good results.

3.1.4 General activity monitoring

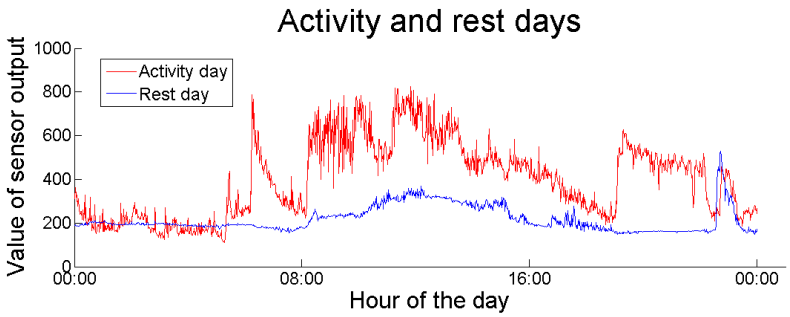
Artificial noses are affected by different environmental factors, such as day and night cycles or activity and non activity periods. For seeing these long cycle components effects is necessary to show long time series of entire days. In such long graphs, different related to seasonality components are clearly seen, but short term cycles are hard to see. Also in such long time series the noise is hard to be seen clearly.

In Figure 3.17 four days, divided into two graphs are represented from 00:00 to 23:59. Red time series are sensor values registered during two working days. Blue ones are values for two resting days. Reactor 1 chemosensor has been used to represent the graph.

In both graphs we can see the difference between a working day and a resting



(a) First example of an activity and a resting day.



(b) Second example of an activity and a resting day.

Figure 3.17: Day time series representing activity and non activity.

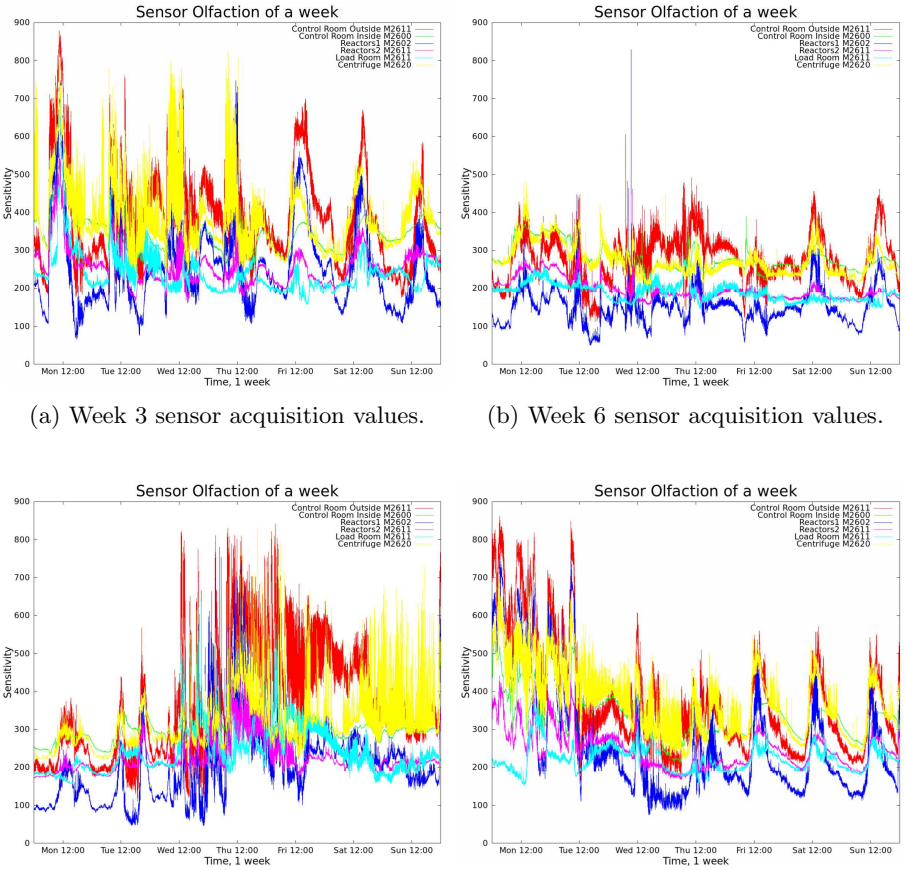
day. We can see an evident difference between factory activity and resting days. While in working days the signal fluctuates and has high values, in the non working days signal is very plain with almost no noise. On first graph, the work starts approximately at 14:00 and when the day ends there is still activity while, on second graph activity day, most work seems to occur from 8:00 to 00:00.

For the four signals, a resting period during nighttime (from 00:00 to 7:00) can be observed in the form of a steady low value in the signal. In activity days during nighttime there is more signal variability. In resting days, signal value raises from 7:00 to 16:00 but with slow gradient, so noses are affected by day cycles even when there is no production.

We can conclude that is easily differentiable when there is activity in the plant and when not because of some reasons. In activity days the media of the signal is higher and the fluctuations have wider range. Also it looks like that, even with activity, the signal drops on night time, but for working days there is still much noise.

Figure 3.18 represents four time series of the six sensors of period three. Each graph lasts a week, starting in Monday at 00:00 and ending on Sunday at 23:59.

The first feature that can be seen in the graphs is day night cycles. During daytime, noses have a higher response than on night time. This change in the output is progressive, causing stationary 24-hour cycles. Different reasons can cause this, the most evident one is that the activity in the factory is higher during daytime, so more odor plumes are present. Other reason is that, during night, temperatures are lower and, when a gas has more temperature, their



(a) Week 3 sensor acquisition values.

(b) Week 6 sensor acquisition values.

(c) Week 7 sensor acquisition values.

(d) Week 8 sensor acquisition values.

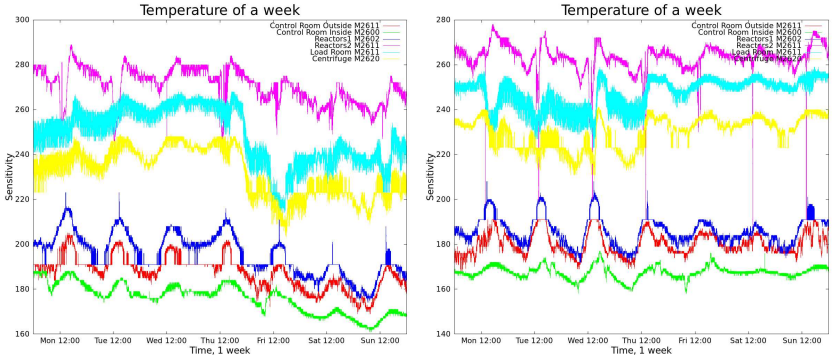
Figure 3.18: Four one week length time series examples of the values acquired by the nose sensors in Elda plant.

reaction with chemoresistors is higher, so during nighttime the noses react less to same airborne substances.

Other interesting characteristic that can be seen at glance is that all the sensors seem like having a correlation. They all have most of the time same ups and downs at almost the same time, so we can see in the graphs that the six artificial noses share the same environment, which means that they are exposed to substantially the same airborne substances and environmental characteristics. Although they practically share the same air, depending on the nose the responses are more or less pronounced. This is sound, because each artificial nose has a kind of sensor that reacts differently to substances, and also is logical to have the same compounds suspended in the air but in different concentrations depending on the place of the plant (because of the air flow present in the place and also the different accessibility level of each area of the factory), so also even having a positive response, the same sensor can react differently depending on the place of the factory where is placed.

As we can see at the beginning of Figure 3.18a, at the end of Figure 3.18c and at the beginning of Figure 3.18d (take in account that those last figures represent consecutive weeks so the end of Figure 3.18c corresponds to the beginning of Figure 3.18d) there are easy to differentiate seasons depending on the activity the artificial noses record. Time series suffer more activity reflecting it in the values registered during that period. This is represented and explained in graph 3.17. Noise variance in activity periods is higher than in the other ones.

Another graphic element visible in Figure 3.18b is the single event which is placed shortly before 12 am on Wednesday. The sensors radically increase their value and then decrease it with high speed too. This is probably due to a compound present in the air for a short period of time that made the noses to react and then go back to its normal values.



(a) Week 5 temperature acquisition values.

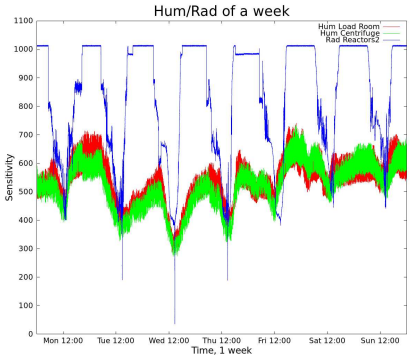
(b) Week 13 temperature acquisition values.

Figure 3.19: Time series examples of the values acquired by the temperature sensors in Elda plant during one week. Some time series are correlated.

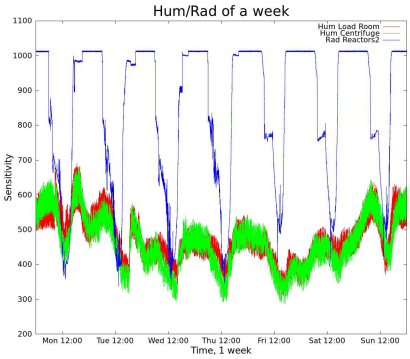
In Figure 3.19 we can see two graphs representing the temperature sensors output during one week each one. These graphics show the chemosensor temperature for each nose. As mentioned previously, sensors react to the same compound in different ways (with more sensibility or less) depending on the sensor temperature, so registering these values can help to better classify all the events. *Control Room Outside* and *Reactors1* have similar time series, the same as *Load room* and *Centrifuges*. Even being these two pairs of time series similar, they don't have the same values because the temperature sensors don't have the output normalized. Artificial noses are built in different stages and not all the components are from the same batch, so probably there are different temperature sensors. Anyways the sensors represent the changes in temperature to take that variable in account when analyzing the data.

In every time series is evident the day night cycles, as can be seen in nose sensor graphs (Figure 3.18). The cycles don't give fixed temperatures for the same hour every day, but the trend is stable. It seems in the graph that some times there are too low values that don't represent the temperature in the environment, so would be useful to define outliers to delete them, or change its

value to a minimum.



(a) Week 1 humidity and radiation acquisition values.



(b) Week 3 humidity and radiation acquisition values.

Figure 3.20: One week length time series examples of the values acquired by the humidity and radiation sensors in Elda plant. Day cycles are clearly differentiated.

In Figure 3.20 we can see two graphs, of one week length each one, representing the two humidity sensors and the radiation sensor time series. As being considered special sensors, and only being three, they were represented in the same graph. Is interesting to remember the reader that the radiation sensor has the output inverted, what means that when higher the value, the lower the radiation that reaches the sensor.

The radiation sensor detects the number of photons that reach the sensible area. The output saturates during night time, what means that is sensing low concentrations of photons, but the sensor is optimized to work with high concentrations. It is focused to record clearly the activity during daytime than in nighttime.

The humidity sensor varies in a less wide range but is more sensible to humidity in the air. This fact appears as noise in the graph. Another fact in the humidity time series is that even being in the Load room and in the Centrifuge area (see Figure 2.10 to check the positions), the two time series overlap, demonstrating that the humidity is almost the same in the whole factory.

Both types of sensors have also day and night cycles as can be seen also in chemosensors time series (Figure 3.19) and in temperatures time series (Figure 3.18) having the minimal values after twelve PM in both types of sensors. Both humidity and radiation signals ascend and descend at the same time demonstrating a correlation, probably because of the light making the water molecules to rise their temperature and forcing them to ascend and leave the plant. So during daytime there is less humidity and more presence of photons.

Sensor reaction to photons presence decrease fast (signal increases) some hours after 12 PM but they increase slowly before. Water molecules presence also rise fast at the same time. The radiation peak is punctual, so the minimum value between day is not continuous. On the other hand, humidity signal is more stable, so water molecules density change slower in the environment. Also

about photon presence peaks, the value is not the same every day, as we can see in the graph in Figure 3.20a. Monday Friday Saturday and Sunday minimum radiation values (which mean maximum photons on the day) are higher than Tuesday Wednesday and Thursday ones. As we know, different days usually have different light. Clouds, weather conditions and the season in which we are can define radioactivity sensor values.

3.2 Automatic event classification

It has been seen before in this chapter that there is not a template that can define any of the events, so any template based classifier will not achieve to detect them. There are no clues on the features that define each event, so a learning process is needed to make the classifier extract them. It must have a supervised learning phase, to focus in the desired events. This classification is hard because of some characteristics the system has: sensor drifting, noise in the signals, events being affected by previous history sensibility and limited number ob events for generating training and test sets. Having these characteristics, classifying events in this environment is a hard task, even supervised learning classifiers can have difficulties, probably giving low accuracy rates, but we expect higher rates than 50%.

Four different events are registered: soda load, methanol load, reactor startup and reactor unload. Both reactor startup and unload are events registered only during the third period while soda and methanol load are registered in all of them. The event labels are given by the workers in the plant, so, as mentioned before, the labels don't give the exact event time.

For making a good classification, many patterns are needed to make the learning process and other few are also needed for testing the capabilities of the classifier. Period 2 was short with almost no activity in the plant, so very few patterns of each event are available. Also it does not acquire different information regarding to period 3, since this last one is defined by the five noses installed during period 2 plus an extra one. Because of these reasons, no classification of events recorded in period 2 is done.

For classifying the events, different based on support vector machines (henceforth SVM) classifiers are used. These different classifiers are a SVM instance, a SVM instance with cross validation (from now on CSVM), a SVM with autoregressive time series prediction (henceforth ARSVM) and a SVM combined with nonlinear dynamical systems (from now on DSVM). As explained in *Data processing* section in Chapter 2, SVM and CSVM are the simplest classifiers while ARSVM and DSVM are more complex ones.

In period 3 there are registered fifteen signals for each event (six chemosensor signals, six temperature ones, two of humidity and one of radiation). Also as being DSVM and ARSVM more sophisticated, the program which implemented it also is more complex, requiring more processing time. Classification of any event for the third period with these classifiers has been impossible due to lack of time and processing resources. On the other hand CSVM and SVM are used for classifying every event during periods 1 and 3.

The classifiers based in SVM use supervised learning, which means that to train them we need a set of classified patterns called training set. Usually when more patterns, more information can be extracted by the classifier. Also another set of classified time series (called test set) is needed for checking the classification capabilities. When more patterns are given in the test set, more accuracy resolution can be given. Is important to not have the same patterns in both sets, so they must be composed of different ones. The events have variable duration, but this is not compatible with the classifiers. To fix this issue and, a fixed event length is defined, fifteen minutes for soda and methanol events and forty minutes for reactor startup and unload. As being more important the beginning of an event than its end, now the events are defined by the beginning

time stamp and a duration.

For extracting the patterns for soda an methanol events, we used the next method: The event patterns are created extracting the time series with the stipulated length (fifteen minutes) from the start of the event marked by the labels. The time series defined by the end of one event and the beginning of the next one (time series extract with no events) are divided into consecutive time series of the defined length, not using the remaining parts that are less long than stipulated. With this method we have more non event patterns that event ones.

For extracting patterns for reactor startup and unload events, other method is used. The event patterns are extracted with the same process as for soda an methanol load events, but with a length of forty minutes. As reactor startup and unload events define the start and the end of the activity periods in the plant, for extracting the non event patterns, a long period of continuous non activity has been cut into consecutive time series of forty minutes long, discarding the last time series with less length.

For checking the variability of the classifiers, for every event and period, few training and test sets are generated. To create them, the patterns are extracted randomly from the previously created database. In period 1 we have registered more events than in the third. Training and test sets in period 1 are composed of two hundred patterns each set while in period 3 only one hundred patterns are given. For making the classifier learn better, both classes have been balanced, so in period 1 each training set has one hundred event patterns and the same number of non event and in period 3 the sets are composed by fifty patterns of each type. The test sets have the same number of patterns of each class than the training sets (one hundred of each type in period 1 and fifty of each type for period 3). As patterns in period 3 have more data than in period 1, for making the classifiers learn fast enough, the sampling rate of the signal in period 3 is one recording each minute while in period 1 recordings are taken each second. When more sampling rate in the patterns, the classifier will be potentially more accurate, but a balance between it and signal learning time is needed.

Having different patterns and signal acquisition rate for each period makes not recommendable to compare the results for the two periods.

One variable is swept to find its best value for each classifier for a better accuracy, the tolerance. The tolerance in four classifiers define when the training is terminated. It ends when the gradient of the optimized function is less than or equal to tolerance. A tolerance value that is too high may cause the SVM algorithm to terminate training before the support vector function is adequately optimized. A tolerance value that is too low will cause the SVM algorithm to try to achieve a very high level of optimization, which may be too time-consuming and computationally expensive and can generate overfitting (try to learn non intrinsic characteristics of the pattern). Overfitting when low tolerances can be reduced adding new patterns to the training set.

For SVM instance, other pair of values are swept, gamma and cost. Gamma value is a variable used for creating the kernels. A high gamma value requires more support vectors to classify the sample, while a less value requires less support vectors. A high value improves the classification for equally distributed feature types while a low value is better when the features are irregularly distributed. Cost specifies the penalty for training errors. High-cost values prohibit training errors, producing a narrow margin and rigid classification while a low-

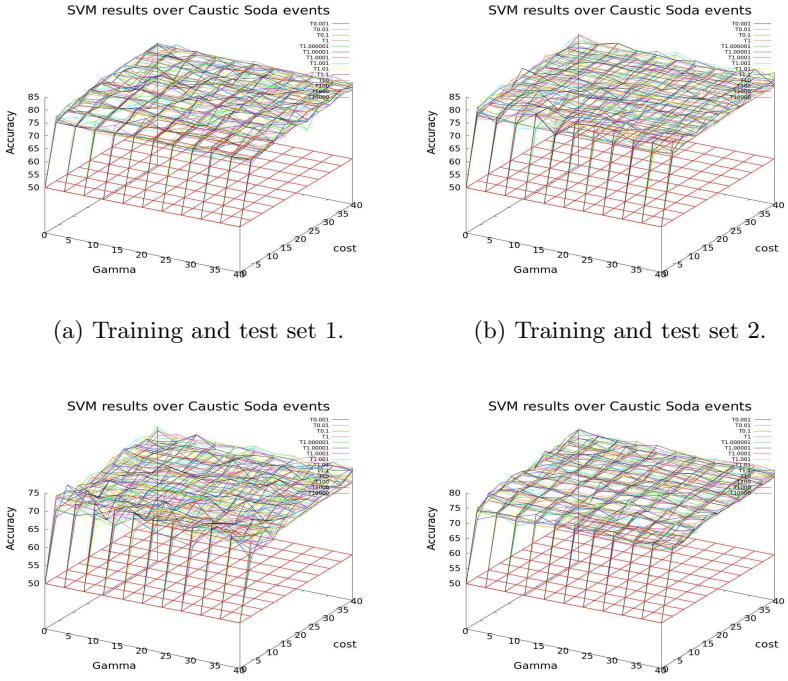
cost value generates a softer margin between classes.

3.2.1 Caustic soda event detection

The following section debates the graphs that represent the success or error rate of the different classifiers for caustic soda load events in the system.

SVM

In Figure 3.21 we can see four graphs that represent the accuracy of the SVM classifier varying tolerance, gamma and cost each one using different test and training sets made with period 1 patterns.



(a) Training and test set 1.

(b) Training and test set 2.

(c) Training and test set 3.

(d) Training and test set 4.

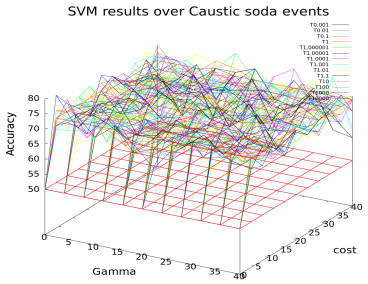
Figure 3.21: SVM classifier accuracy as a function of tolerance, gamma and cost variation for soda load events in period 1. The variables don't affect too much the accuracy.

Glancing at the four graphs we can realize that the different trained classifiers have similar accuracy. Values range from 70% to 80% success rate. These values significantly improve the null classifier (it consists in giving all patterns the same label) which gives an accuracy of 50%. Also can be shown in the graphs that the shapes generated are similar for each pattern. Varying gamma value between 0 and 40 does not change the accuracy of the classifier, while changing the cost also makes no effect too on values above 10. For cost 0 the classifier is as bad as the null classifier and it quickly reaches the highest values at cost 10,

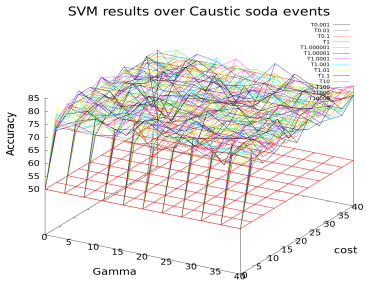
observing a little ascend from cost 5 to 10. Varying the tolerance also seems to have no effect in the classification capability except for values of 2 and more, which makes the classifier capabilities drop to 50%. Varying gamma, cost and tolerance makes the classifier classify with an accuracy that varies five points approximately from best to worst value, demonstrating that the classifier has not a stable classify strategy.

So the best values for the SVM classifier for caustic soda events for period 1 are tolerance 0.9 cost 10 and gamma 1 with a median accuracy of around the 75%. These are the best values instead of other higher because those are the ones that require less processing for the same classification accuracy. Tolerance value is defined in 0.9 because of suffering a non equilibrium point when used values similar to 1. This is seen in graph 3.23b and can be applied to SVM because CSVM is formed executing repeatedly SVM over different learning process based in the same training and test sets.

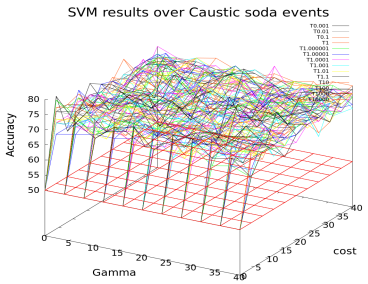
In Figure 3.22 four graphs represent the accuracy percentage for different instances of the SVM classifier applied on the caustic soda load events in period 3 varying cost, gamma and tolerance.



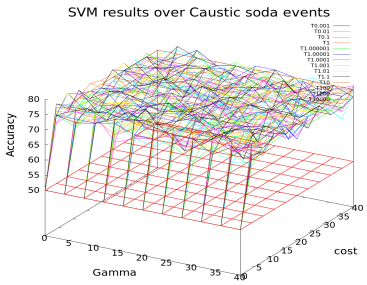
(a) Training and test set 1.



(b) Training and test set 2.



(c) Training and test set 3.



(d) Training and test set 4.

Figure 3.22: SVM classifier accuracy as a function of tolerance, gamma and cost variation for soda load events in period 3. More variability in accuracy is present than in period 1.

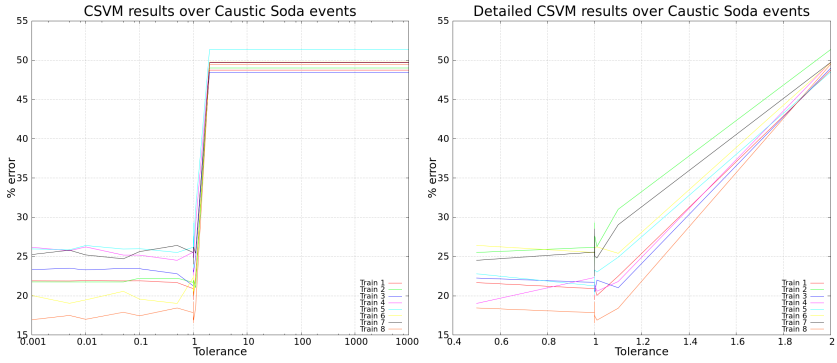
The accuracy of four examples show values between 70% and 80%. In these graphs we can also observe the same pattern as seen with SVM classifier over

caustic soda load events in period 1 (Figure 3.21). The cost values make the classification accuracy raise from 50% to around 75% depending on the training and test sets used, the variation of the cost over 10 makes no change in the capabilities of the classifier. The gamma value don't change the success rate and the tolerance values over 10 make the classifier give a 50% of accuracy. The variability of the success rate reflected in small changes of tolerance, gamma or cost are higher than in the same classifier in period 1. Probably this is due to having simplified the time series sampling rate to only 1 point each minute.

Again the best values for the SVM classifier for caustic soda events for period 3 are tolerance 0.9 cost 10 and gamma 1 with a median accuracy of around the 75%.

Cross SVM

In Figure 3.23 we can observe two graphs reflecting the variability on the error percentage of the CSVM classifier for caustic soda event detection in period 1 for tolerance values between 0 and 1000. X axis has a logarithmic scale representation. Figure 3.23b shows accuracy for tolerance values between 0.5 and 2 with arithmetic scale representation of both axis. This scale is present in all graphs that represent CSVM error rate.



(a) Error of CSVM soda classifier in period 1.

(b) Detail of error of CSVM soda classifier in period 1.

Figure 3.23: CSVM classifier error rate as a function of tolerance variation for soda load events in period 1. There is a turning point for tolerance equal to 1.

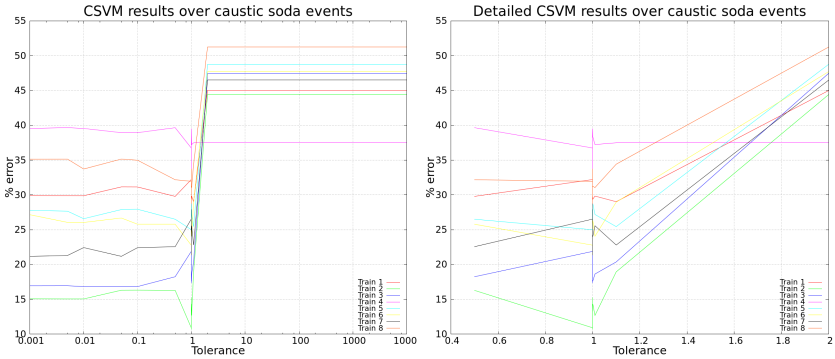
The error rates of this classifier are between 15% and 25% which leaves an accuracy of 75% to 85% depending on the experiment.

Even having similar behaviors, different training and test sets allow the classifier different error rates. The error values remain in a specific value (from 16% to 26% depending on the training and test sets used) with little variance for tolerance 0 to tolerance 1. For tolerance values above 2, the classifier capabilities decrease leaving its error rate to near 50% remaining stable in this value. When tolerance equals to 1 there is a change in the trend ascending and descending its error rate in almost the same tolerance value, so this is a non equilibrium point that is interesting to evade. For tolerance values between 1 and 1.2 there is a

small valley with a minimum error rate that defines the best tolerance value for the classifier. The problem is that depending on the training and test set, this valley is closer to tolerance 1 than to tolerance 1.2 and after the valley the error rate ascends with a high gradient, so not being able to define the exact point where it appears can make the classifier lose capabilities.

The best tolerance for classifying with CSVM for soda load events in period 1 is 0.9, being a number near 1 but evading the variability of the error rate at values near 1.

In Figure 3.24 two graphs reflect the variation in the error percentage of the CSVM classifier for caustic soda event detection in period 3.



(a) Error of CSVM soda classifier in period 1.

(b) Detail of error of CSVM soda classifier in period 1.

Figure 3.24: CSVM classifier error rate as a function of tolerance variation for caustic soda load events in period 3. There is a high error difference between examples.

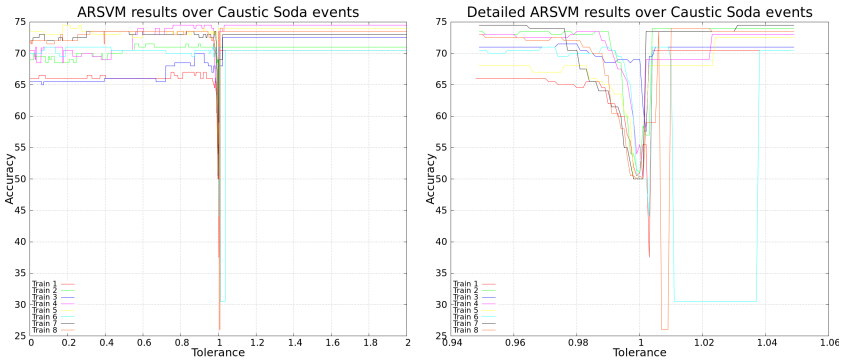
The error rates of this classifier are between 15% and 40% so the success rate has a value of 60% to 85% depending on the training and test sets.

The error rates between experiments have wider values, having a difference between the best and the worst accuracy at tolerance 0.8 of 25%. This is a great difference for being the same classifier with different training and test sets, so we can determine that the stability of CSVM in period 3 for caustic soda load events is low. This is probably because of the sampling rate of one registration per minute. For tolerance minor to 1, in each example, the error rate is stable with a small variability of not more than a pair of points from the higher value to the lower. On tolerance 1 we have the same non equilibrium point as in CSVM for soda in period 1. Also the valley with the minimum error rate between values 1 and 1.2 is still present for almost all the experiments. After this point the accuracy descends to values near 50%, having more range of values being most of them between 45% and 50%.

The best tolerance value for CSVM classifier to classify caustic soda load events in period 3 is 0.9, being a number near 1 but evading the variability of the error rate at these values.

Autoregressive SVM

In Figure 3.25 two graphs reflect the variability in the accuracy of ARSVM classifier for caustic soda load events with tolerance values between 0 and 2 with normal representation of both axis and a zoom of the values around tolerance equal to 1.



(a) Accuracy of ARSVM classifier.

(b) Detail of accuracy of ARSVM classifier.

Figure 3.25: ARSVM classifier accuracy as a function of tolerance variation for caustic soda load events in period 1. For tolerance 1 there is a decrease in accuracy.

Best success rates have values between 70% and 75% for all the examples, being the variability range between examples reduced to less than 5 points.

As all the lines are close one each other, we can define that the classifier is very stable regardless of the different examples taken for the training, what makes us understand that the classifier grabs easily a group of significant features in the time series. Three value intervals in tolerance variable can be defined depending on the accuracy. The first interval goes from values 0 to 0.95 and is characterized by an stable accuracy with a variability range of few points for the same experiment and less than ten points between different experiments. The second interval is for tolerance values near 1. Almost all the lines suffer a descent in the accuracy forming a valley and then raising again their values reaching similar success rates before and after the valley, being slightly higher for values over 1. Most valleys start at tolerance 0.98 and end at tolerance 1.01 but some examples have the valley at values over 1. The last interval is for tolerance values over 1 and is defined by a constant accuracy independently of the value of the variable. A minimum range of five points in the success rate is seen between the best and worst values in this interval.

After analyzing the graphs, the best tolerance values for the ARSVM classifier for caustic soda load events in period 1 with the fewest mistakes are values above 1.2, being a good option 1.5 to ensure that the value is not inside the valley.

Dynamic SVM

Figure 3.26 contains a single graph that represents the accuracy of DSVM classifier with tolerance variation for soda load events for different training and test sets for tolerance values between 0 and 2.

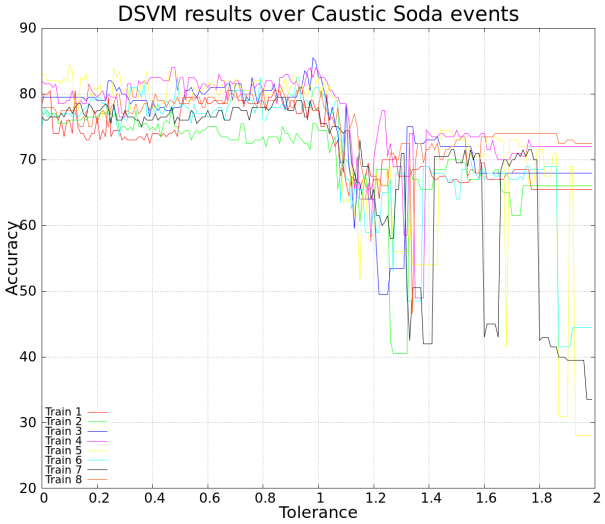


Figure 3.26: DSVM classifier accuracy as a function of tolerance variation for caustic soda load events in period 1. Accuracy is stable until tolerance reaches values above 1, then it decreases.

The best results given for this DSVM classifier give success rates of 75% to 85% at values near 1 being below this number.

In the graph we can see that tolerance values less than 1 tend to raise as they increase, having a maximum in almost all examples at tolerance 0.95. The variability that suffers the accuracy between tolerance 0 and 1 is of a few points, not being much higher than it suffers between different examples, never reaching the difference of ten points. This fact makes the DSVM classifier very stable independently of the training chosen for its learning process. When tolerance is bigger than 1, the success rate descends in different gradients depending on the example reaching values near 40% for some cases, what has not too much sense, because being balanced classes, the dumbest classifier can have a 50% accuracy and is irrational to reach lower values.

So we can define after checking the results that for DSVM classifier for caustic soda load events in period 1 the best tolerance to be defined is near 1 but below this value, such as 0.95.

Classifying conclusions for caustic soda load event detection

So for classifying of caustic soda load events for period 1 we have the next results: SVM reaches values of 70% to 80% , CSVM has success rates between 75% and 85% , ARSVM has an accuracy of 70% to 75% and DSVM is between 75% and 85%. There would be no difference in using CSVM and DSVM because of having similar success rates, but taking another look to graphs in figures 3.26

at tolerance value 0.95 and 3.23b at tolerance value 0.9 we can see that the examples have wider range in CSVM than in DSVM, so the best option would be to use DSVM with tolerance value of 0.95.

For classifying of caustic soda load events for period 3 we have the next results: SVM have a success rate between and 70% and 80% , while CSVM is placed with values between 60% and 85%. Clearly the best option here is to use SVM with tolerance 0.9 cost 10 and gamma 1.

In both SVM classifiers for soda load, with six noses we have more noise when varying the value of the variables but the median of the accuracy is the same. Also is interesting to know that while period 3 has eight times the number of time series it has sixty times less points defining the series.

For CSVM comparison between periods 1 and 3, both have the same highest accuracy percentages, but in period 3 the median of success rate is lower, a 72.5% against a 80% for period 1. Having more time series per example in period 3 but less sampling rate in time series does not increase the accuracy, but makes the classifier less robust depending on the examples chosen.

3.2.2 Methanol event detection

In this section we are discussing the results of the four classifiers for methanol load events for periods 1 and 3.

SVM

In Figure 3.27 we can see the accuracy of four SVM classifiers trained and tested with different sets with variation in tolerance, gamma and cost.

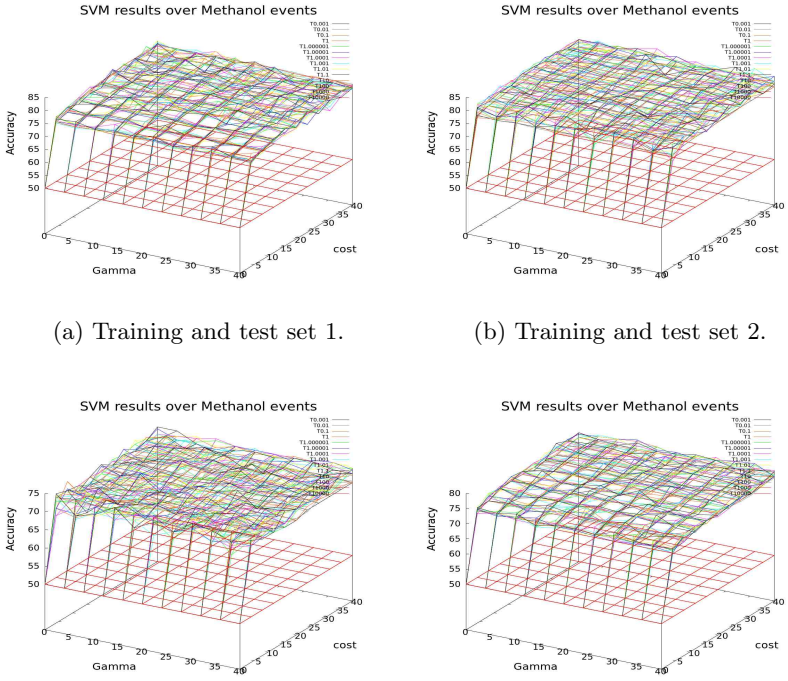
As being common in the four graphs, the best values are between 70% and 80%, being an stable maximum for a wide range of three variables. For tolerance, for any value lesser than 2 the classifier classifies with almost no variability in the accuracy , while for 2 or higher, the classifier becomes a null classifier. For gamma the classification has similar success rate, independently of its value, and for cost the same occurs for values of 10 and above. For cost 0, the classifier does not make any classification, but gives all the patterns the same label, improving quickly the success rate reaching the maximum at cost equal to 10. The results are not much different compared with SVM classifier applied on soda load events for period 1, as can be seen in Figure 3.21. The noise in accuracy through changing tolerance, cost and gamma values is of around 10 points, slightly higher than for the SVM classifier in soda events.

The results of this classifier give us an accuracy of 70% to 80% for tolerance 0.9, cost 10 and gamma 1. These values again are the best because they are the ones that make the learning process require less time giving the maximum accuracy values.

In Figure 3.28 four graphs represent the accuracy percentage of different instances of the SVM classifier applied on methanol load events varying cost, gamma and tolerance.

The maximum accuracy given for this classifier through the four examples are between 65% and 80% for a high range of tolerance, cost and gamma values.

The classifier shows the same response to tolerance, gamma and cost variation as in period 1 (Figure 3.27) with the difference that noise, when making small changes in the variables, rise to almost fifteen points. Again, tolerance



(a) Training and test set 1.

(b) Training and test set 2.

(c) Training and test set 3.

(d) Training and test set 4.

Figure 3.27: SVM classifier accuracy as a function of tolerance, gamma and cost variation for methanol load events in period 1. Accuracy decreases for cost equal to 0 and tolerance greater than 2.

values above 2 make the classifier useless, but it gives maximum success rate for tolerance minor than 2. For cost the same happens, it gives an accuracy of 50% when cost is 0, independently of the other values, and rising to the maximum success rate at 10 , being stable this output for any value above. For gamma, the value it takes is irrelevant and does not affect the result of the classifier.

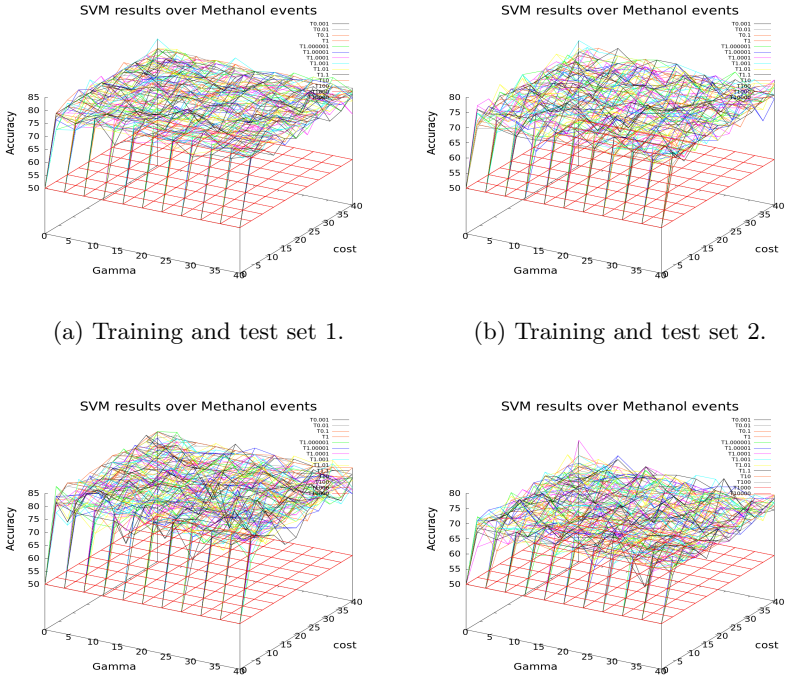
After analyzing the results of SVM classifier for methanol load events detection during period 3, the best result comes when tolerance takes a value of 0.9 with cost 10 and gamma 1. Again as in period 3, these values give the higher success rate with the less process consumption.

Cross SVM

In Figure 3.29 two graphs reflect the variability in the error percentage of the CSVM classifier for methanol event detection in period 1.

The classifier has a minimum error rate between 17% and 27% for tolerance value of 0.1 through the eight examples painted in the graph. What means that the classifier reaches success rates from 73% to 83%.

The reaction of the classifier for different values of tolerance shares the same characteristics of CSVM for soda load event detection (Figure 3.23). For tolerance values from 0 to below 1, the classifier has an error rate between 15%



(a) Training and test set 1.

(b) Training and test set 2.

(c) Training and test set 3.

(d) Training and test set 4.

Figure 3.28: SVM classifier accuracy as a function of tolerance, gamma and cost variation for methanol load events in period 3. As for caustic soda in period 3, SVM shows no variability in accuracy for small changes in variables.

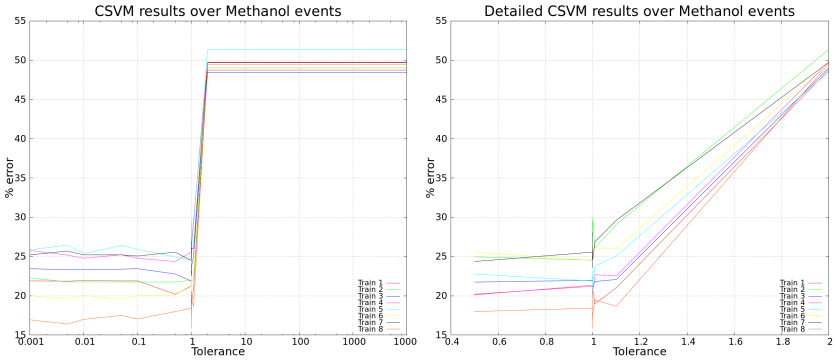
and 27% with a variability of less than five points in the same example and 15 between them. When tolerance reaches value 1 and nearby, a non equilibrium point appears making the error rate fluctuate higher and lower than the media. For values above 1, the classifier increases its error rate reaching values around 50% when tolerance equals to 2, making it a null classifier for higher tolerance values.

As discussed, the best tolerance value for CSVM classifier for methanol event detection in period 1 seen through these examples is 0.1, but the error rate is similar for tolerance values between 0 and 0.9.

Figure 3.30 shows two graphs reflecting the variability in the error percentage of the CSVM classifier for methanol event detection in period 3.

The lowest error rate offered by this classifier for this kind of events goes from 22% to 35%, what means that its accuracy is of 65% to 78%.

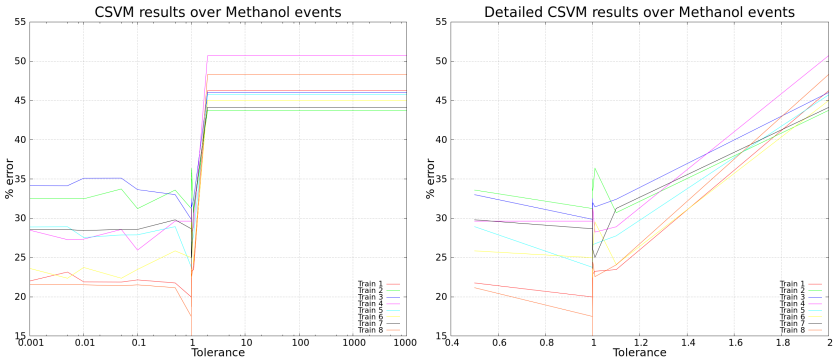
In Figure 3.30a, the graph shows again three periods depending on tolerance values.One period goes from 0 to 1, other from 1 to 2 and the last one from 2 to 1000. In the first period, the values are lower than in the other two, having an error from 20% to 35%, making fluctuations of less than five points for the same example and never more than fifteen points between examples. This range is wider than the same classifier with period 1 patterns. For tolerance values



(a) Error of CSVM methanol classifier in period 1.

(b) Detail of error of CSVM methanol classifier in period 1.

Figure 3.29: CSVM classifier error rate as a function of tolerance variation for methanol load events in period 1. For values above 2, the classifier has high error rates.



(a) Error of CSVM methanol classifier in period 3.

(b) Detail of error of CSVM methanol classifier in period 3.

Figure 3.30: CSVM classifier error rate as a function of tolerance variation for methanol load events in period 3. Error rate difference between examples can reach 15%.

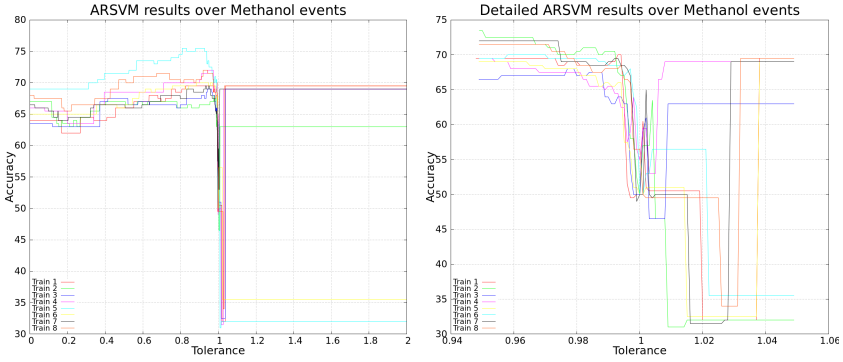
from 1 to 2, the error raises to near 50% while in last period (tolerance greater than 2) the error is stable in the worst possible values, with a variability of a maximum of 7 points between examples. When looking deeper in tolerance values around 1 (Figure 3.30b), there exists the non equilibrium point around tolerance 1. When approaching this value from the left, there is a reduction in the error rate for all the examples and when leaving values near 1 to greater ones, some examples reduce their error rate and others increase it.

The tolerance with the minimum error rate for CSVM methanol event detection in period 3 is a value near 1 without reaching it, but is not interesting

to use these values because of having the non equilibrium point. So, again, as in CSVM applied over period 1, the best tolerance value for having the best accuracy is 0.1.

Autoregressive SVM

In Figure 3.31 two graphs show the variability in the accuracy of ARSVM classifier for methanol load events with tolerance values between 0 and 2 with normal representation of both axis and a zoom of the values around tolerance equal to 1.



(a) Error of ARSVM accuracy classifier in period 1.

(b) Detail of accuracy of ARSVM Methanol classifier in period 1.

Figure 3.31: ARSVM classifier accuracy as a function of tolerance variation for methanol load events. For tolerance values over 1.05, accuracy has a stable high value for some examples and a low value for others.

Best success rates have values between 65% and 75% for all the examples being the variability range between them around 10 points.

In the graphs we can see that for tolerance from 0 to 0.95 the success rate rises slightly and constantly for all the examples, not having more than ten points of difference between the best and worst accuracy. When approaching to values close to 1 but minor, the accuracy has a negative gradient reaching 50% and for values higher than 1 the success rate vary, reaching an stable point for tolerance values higher than 1.05, depending this value on the training set. The accuracy on those stable point vary depending on the patterns used for training, going from 30% to 70%.

The best tolerance for CSVM classifier over period 1 for methanol load event detection is of 0.9, having an accuracy of around 70%.

Dynamic SVM

Figure 3.26 contains a single graph that represents the accuracy of DSVM classifier with tolerance variation for methanol load events for different period 1 training and test sets for tolerance values between 0 and 2.

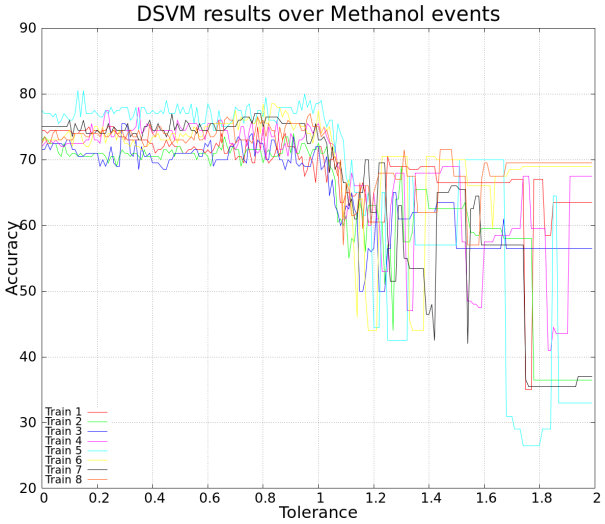


Figure 3.32: DSVM classifier accuracy as a function of tolerance variation for methanol load events. Accuracy remains higher than 70% for tolerance values minor than 1.

The best accuracy reached by DSVM classifier in period 1 methanol events goes from 70% to 80%. The tolerance range for these success rates is wide, going from 0 to 1.

For tolerance 0 to 1, the classifier has its best results having a variability of 10 points between examples and a variability of a pair of points for the same training and test sets. When reaching value 1, the accuracy decreases in a continuous way, but when values above 1.2 are set, the success rates have different punctual jumps for every training and test sets, having a variability of more than thirty points using the same examples and values from 25% to 70% for the whole examples. So the classifier gets really useless for tolerances over 1.2 and also very unstable. But for values below 1, the classifier is very stable.

As conclusion, DSVM classifier in period 1 for detecting methanol load events has as its best tolerance value 0.1 giving results between 70% and 80% and demonstrating a good stability independently of the training set used.

Classifying conclusions for methanol load event detection

Using different classifiers to detect methanol load in the system in period 1 gave next results: SVM with an accuracy going from 70% to 80%, CSVM success rate from 73% to 83%, ARSVM values between 65% and 75% and DSVM a ratio going from 70% to 80%. The best classifier is CSVM with a tolerance value of 0.1.

For methanol load event detection in period 3, the best results are as follow: SVM has an accuracy from 65% to 80% while CSVM have success rates between 65% and 78%. So we can conclude that, for detecting methanol events, the best classifier is SVM with tolerance 0.9, cost 10 and gamma 1. Probably these results would change if more sampling rate is inserted to the time series, because CSVM

only makes a cross validation over SVM rising the success rate for most cases, but not this one.

For SVM classifier for periods 1 and 3 in methanol load detection, the response of the classifier is the same for the different variables, while the maximum success rate is the same but, for period 3 we have more noise in the success rate, probably because of having reduced pattern numbers and only one point registered each minute.

Applying CSVM classifier for methanol load events detection shows similar reactions to tolerance variation even knowing that the time series have different characteristics. Even so, some interesting elements differ between classifiers. The accuracy range for tolerance from 0 to 1 is wider for period 3 than the same classifier with patterns of period 1. Exactly the same happens for tolerance values greater than 2, for period 3 the variability in the error rate between examples is bigger. Having less sampling rate seems to be the reason of this circumstance. Is interesting to realize that, for methanol events, the CSVM classifier does not have a valley for tolerance between 1 and 1.2 as it had in both CSVM classifiers in caustic soda event detection. Instead of being a valley, the success rate rises with clearly less gradient than values above 1.2.

3.2.3 Reactor startup event detection

In this section we discuss the results of the SVM and CSVM classifiers for reactor startup events for period 3.

SVM

In Figure 3.33 four graphs are present representing each one the accuracy of SVM classifier for reactor startup events with different training and test sets.

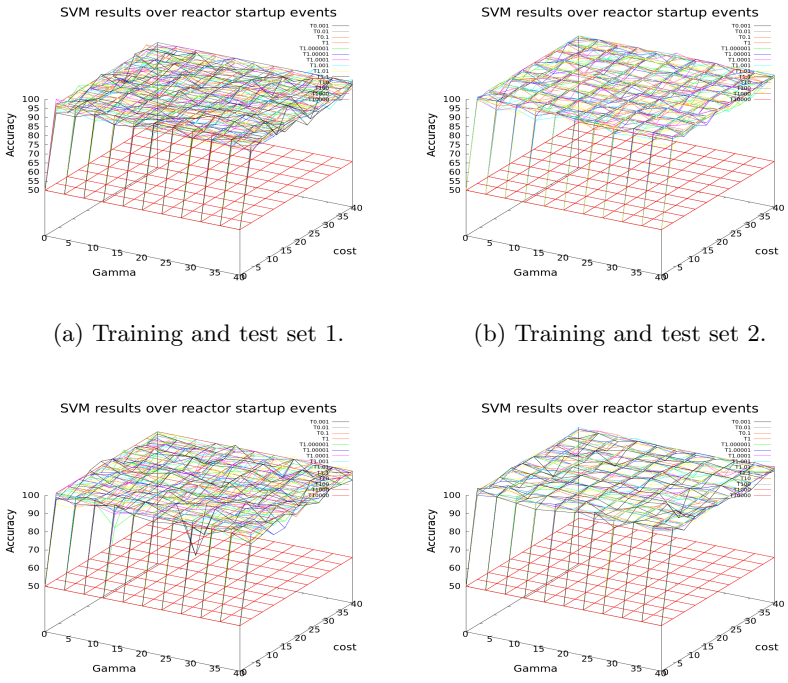
The best accuracy values that the graphs show us are around 100% varying its value between 95% and 100% for a wide range of tolerance cost and gamma values.

The three variables present in the graph show very little changes in the success rate of the classifier. For tolerance of 2 or higher the classifier equals a null classifier with 50% success rate, while for values lower than 2, it classifies at its maximum success rate (more than 95%). For cost equal to 0, it gives 50% of accuracy while it ascends to the maximum success rate from value 0 to 10. For cost values over 10 there is no change in accuracy, being the maximum. Independently of gamma value, the classifier has its highest values. There are fluctuations on the graph for variable changes making accuracy variate from 95% to almost 100%.

For using SVM classifier in period 3 for detecting startup events, the best variable values are tolerance 0.9, cost 10 and gamma 1 for making the classifier learning as faster as possible without sacrificing success rate and giving success rates between almost 0% and 5%. We try to avoid giving tolerance a value of 1 because of having a non equilibrium point as can be shown in Figure 3.30b.

Cross SVM

In Figure 3.30 we see two graphs that reflect the variability in the error percentage of the CSVM classifier for reactor startup event detection.



(a) Training and test set 1.

(b) Training and test set 2.

(c) Training and test set 3.

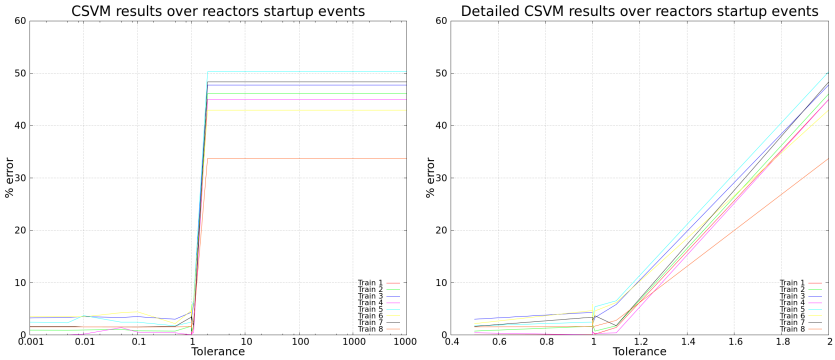
(d) Training and test set 4.

Figure 3.33: SVM classifier accuracy as a function of tolerance, gamma and cost variation for reactor startup events in period 3. Maximum accuracy reaches values near 100%.

The lowest error rate reached by CSVM classifier in this case is of almost 0% to 3% at a tolerance between 0 and 1, giving an accuracy of 97% to near %100.

When tolerances are below 1, the success rate through all the examples goes from 0% to 4% having a variability in the same example of less than 2 points and between them of less than four. When tolerance has a value of 1, there is a non equilibrium point raising and decreasing chaotically the success rate. Distancing from tolerance 1 to tolerance 2, the classifier decreases its success rate until it reaches values near 50%. But, as can be seen in Figure 3.34b, for values slightly higher than 1, some training sets make the classifier to enhance or worsen the success rate with no apparent patron. For tolerances higher than 2, the success rate does not vary, having values between 35% and 50% depending on the training set used for making the learning process.

The best tolerance value for CSVM classifier for detecting startup events is 0.5 giving an error rate between almost 0% and 3%, resulting in an accuracy between 97% and almost 100% Having bigger training sets can reduce this success rate variability even though the results have very high accuracy.



(a) Error of CSVM startup classifier in period 3.

(b) Detail of error of CSVM startup classifier in period 3.

Figure 3.34: CSVM classifier error rate as a function of tolerance variation for reactor startup events in period 3. Almost no error is present for tolerance values minor to 1.

Classifying conclusions for reactor startup event detection

For the classifiers applied for reactor startup event detection we have the next results: SVM gives an accuracy of 95% to 100% while CSVM goes from 97% to near %100. The best classifier for detecting these kind of events is CSVM with a tolerance of 0.5 resulting in great accuracy.

3.2.4 Reactor unload event detection

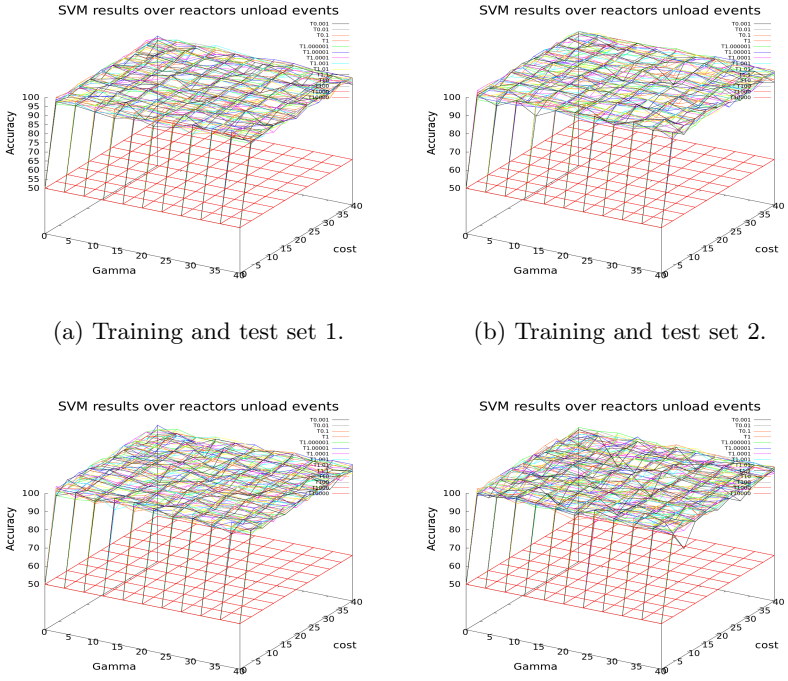
Forward we are discussing the results given by the SVM and CSVM classifiers for the reactor unload event detection during period 3.

SVM

Figure 3.33 contains the graphical representation of four training and test sets of reactor unload events over period 3 for making the learning process for SVM classifier with different values for tolerance, gamma and cost.

Watching the graph we can say that the clear best performance goes from 95% to 100% over most values of tolerance, gamma and cost.

As reflected every time the SVM classifier is used, gamma, in the whole range of values used, does not change the accuracy, leaving it in its maximum. While tolerance makes no change in the accuracy for values lower than 2 giving maximum success rate, for values of 2 or above, makes the classifier act as the null classifier, giving every pattern the same label with an accuracy of 50%. Finally, cost, when is 0, makes the classifier give a 50% of accuracy but, from 0 to 10, success rate rises from this value to a maximum near 100%, remaining stable for cost values over 10. There is a variability of a few points when making small changes in cost, gamma or tolerance beneath the values that keep the classifier with maximum success rate.



(a) Training and test set 1.

(b) Training and test set 2.

(c) Training and test set 3.

(d) Training and test set 4.

Figure 3.35: SVM classifier accuracy as a function of tolerance, gamma and cost variation for reactor unload events in period 3. Accuracy reaches near 100% values as for reactor startup events.

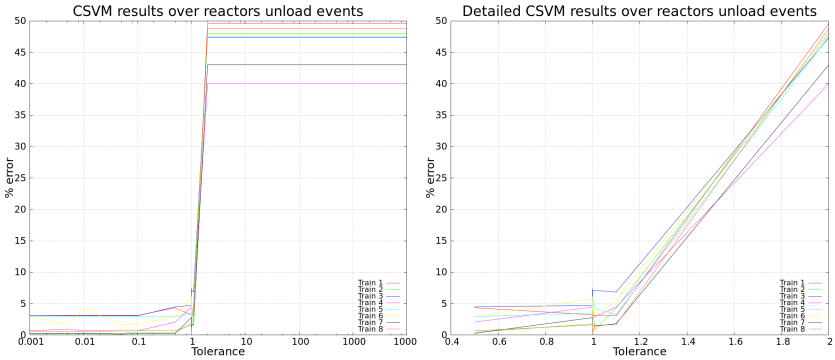
The SVM classifier trained with different training and test sets of reactor unload events during period 3 give an accuracy from 95% to almost 100% with tolerance equal to 0.9, cost with a value of 10 and a gamma of 1. Again this values are chosen for being the values that, giving the maximum success rate, are the ones that makes the learning process for SVM easier to learn.

Cross SVM

In Figure 3.30 we can observe two graphs reflecting the variability in the error percentage of the CSVM classifier for tolerance values between 0 and 1000 for the reactor unload events.

A global view over the graph shows that the lowest error rate achieved by this classifier goes from 0% to 3%, what means a 97% to 100% success rate for tolerance values between 0 and 0.9.

For tolerance values above 2, the classifier has fixed error rate from 40% to 50% depending on the training set used. For tolerance 1 we still have the characteristic non equilibrium point that is present in all the CSVM instances, what makes the error rate vary ascending and descending chaotically for all the examples. For values slightly higher than 1, depending on the training set used in the classifier, the error rate ascends or descends a few points for, afterwards,



(a) Error of CSVM unload classifier in period 3.

(b) Detail of error of CSVM unload classifier in period 3.

Figure 3.36: CSVM classifier error rate as a function of tolerance variation for reactor unload events in period 3. The error rate for each tolerance is almost the same that for reactor startup events with CSVM.

ascending dramatically until reaching tolerance 2. For values between 0 and 1, the classifier gives its best results, having values between 0% and 3% with a variability of few points depending on the tolerance and the training set, but being very stable. CSVM classifier shows robustness for this kind of event detection.

After analyzing the results for CSVM reactor unload event detection, we can say that the best tolerance can be set in 0.5, being this the intermediate point between 0 and 1 and also having what seems a minimum in error rate. This gives a success rate of 97% to 100%.

Classifying conclusions for reactor unload event detection

As commented before, we have the next results: for SVM classifier, accuracy ascends to values from 95% to 100%, while CSVM has more stability in best results, achieving values between 97% and 100%. Even though it never reaches the 100%, the values are very near, reaching sometimes an error rate of 0.03% what means a very high accuracy. We can define that CSVM with a tolerance value of 0.5 is a valid classifier for these kind of events.

3.2.5 Daily cycle detection

During the development of the work exposed in this document, a new capability for the system emerged, maybe it can detect when the factory is active or if it is in a non activity period. Moreover it may detect the quantity of biodiesel made in the plant through the signals registered. So a simple test for period recognition has been done over only period 3.

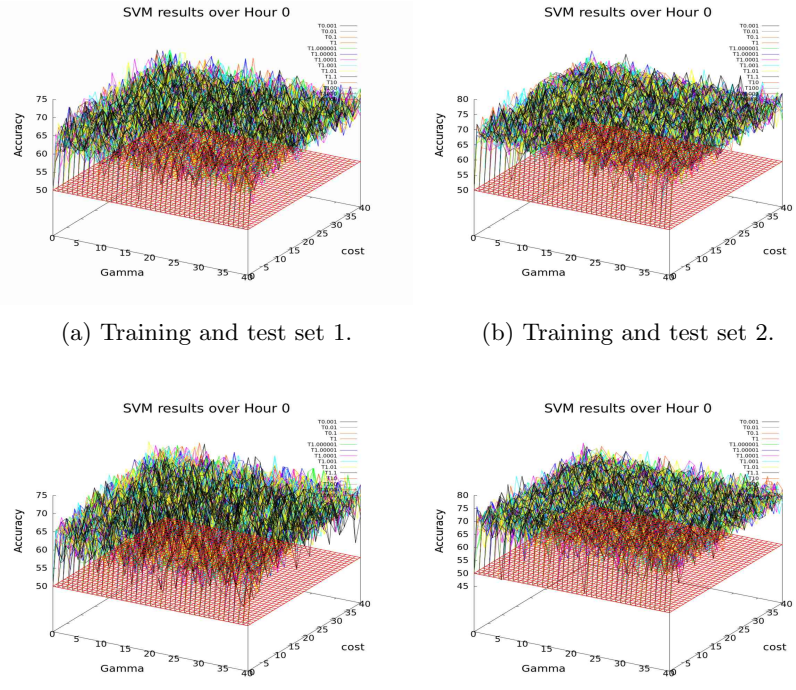
It consists on checking if the four classifiers (SVM, CSVM, ARSVM and DSVM) can detect approximately what time is it only having the signals recorded as input. For this purpose, the fifteen signals of period 3 have been divided into time series of one hour length, starting at 00:00. Different days are used having

twenty four kinds of time series, one from 00:00 to 00:59 (from now on called Hour 0), other from 1:00 to 1:59 (henceforth Hour 1) and so on. For testing if the classifiers are capable of distinguishing in which hour the time series is recorded, Hour 0 has marked as event (labeled as 1) and the rest of patterns are marked as non event signals (labeled as 0).

Having the patterns database, different test and training sets are generated grabbing randomly for each set forty patterns, twenty of each class. The sampling rate of the signal is of one acquisition every second. The fifteen sensor signals of period 3 are included. As commented before, making the classifiers to learn in such conditions take long. In fact the execution of all instances took days in a cluster with fifty parallel microprocessors, so the process cost of this experiment was expensive.

SVM

The first classifier to learn from training and test sets for detecting the Hour 0 over period 3 is the SVM. Figure 3.37 represents four instances of this classifier for these events with different training and test sets used in each one.



(a) Training and test set 1.

(b) Training and test set 2.

(c) Training and test set 3.

(d) Training and test set 4.

Figure 3.37: SVM classifier accuracy as a function of tolerance, gamma and cost variation over the time hour 00:00-1:00 against the rest of hours. There is too much variability in accuracy, probably because of having few training examples.

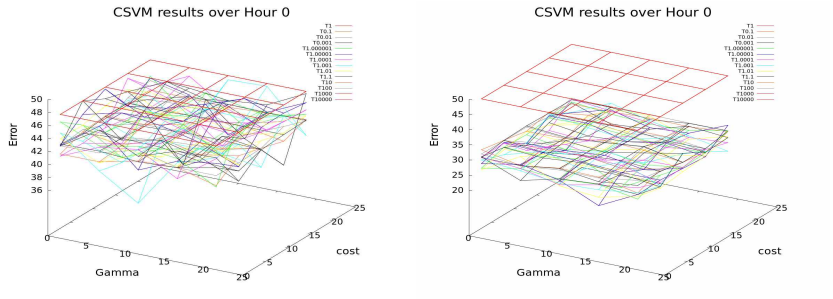
An overview of the graphs gives us in first instance values between 55% and 75% as the best accuracy reached by this classifier.

A more in deep view show us that accuracy of SVM classifier again show the same responses to different variable values. For tolerances above 2, the classifier gives an accuracy of 50% while for lower values it gives the maximum success rate. For all gamma values used in this experiment the accuracy does not suffer any significant change. And, finally, for the cost variable, again for value 0, the classifier acts as a null classifier with a 50% hit rate and, for values above 10, it gives a success rate between 55% and 75%. Values from 0 to 10 make the signal ascend quickly from 50% to the maximum hit rate. Also, for small changes in the three variables, a variability in accuracy is produced of almost fifteen points, demonstrating that this classifier has not much stability for this kind of event detection.

The best results are given through a wide range of values for three variables, but the values that allow the fastest learning process without losing hit rate are 0.9 for tolerance, a gamma of 1 and a cost value of 10.

Cross SVM

As being this experiment a test we have also included in CSVM cost and gamma variation besides tolerance. So in Figure 3.37 we can observe two graphs representing a pair of examples of training and test sets generated for detecting the Hour 0 event with CSVM classifier with tolerance, gamma and cost variation.



(a) Error of CSVM classifier with train 1.

(b) Error of CSVM classifier with train 2.

Figure 3.38: CSVM classifier error rate as a function of tolerance, gamma and cost variation over the time hour 00:00-1:00 against the rest of hours. High error rate differences can be distinguished between examples.

The error rate has high fluctuations depending on the patterns used for training, having in the left graph an error rate media of 42% and in the right a media of 30%, so is hard to define the error, but we can set it between 30% and 42%, which gives an accuracy of 58% to 70%.

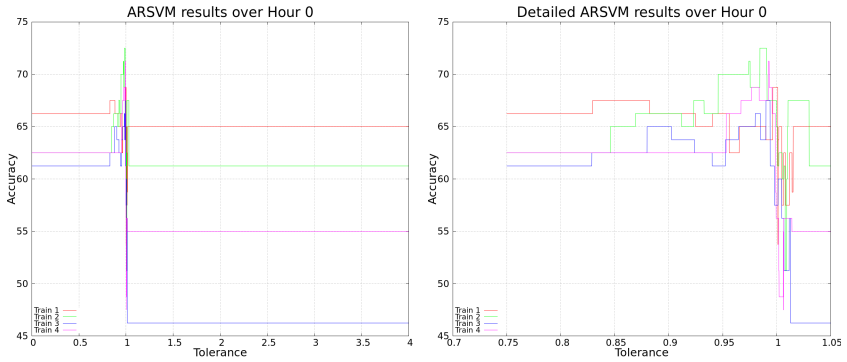
The variation of gamma and cost values give no apparent change in the classifier, maintaining the accuracy in its maximum value except for maybe cost 0 that seems like it decreases the accuracy a few points. For tolerance it acts exactly the same as in SVM, for values of 2 and above, the classifier acts as a null classifier giving 50% hit rate while, for values lower than 2, the response is the maximum independently of the value. There are fluctuations when variating

slightly the three values making the accuracy vary few points, and depending on the training set used.

The best values for CSVM classifier for detecting Hour 0 events in period 3 can be defined as with SVM, tolerance 0.9 gamma 1 and cost 10 giving an accuracy of between 58% and 70%. It is noticeable that, for having a better view of this classifier, more patterns would be interesting to be included in the training set for evading accuracy variability. Also more training sets should be created.

Autoregressive SVM

In Figure 3.39 two graphs show the accuracy of ARSVM classifier for Hour 0 event detection over period 3 for tolerance values between 0 and 4.



(a) Accuracy of ARSVM classifier. (b) Detailed view.

Figure 3.39: ARSVM classifier accuracy as a function of tolerance variation over the time hour 00:00-1:00 against the rest of hours. Maximum accuracy values are present around tolerance value of 0.97.

A view at both graphs determine that the max accuracy that this classifier can give us varies between 65% and 73% for a short range of tolerance values around 0.97.

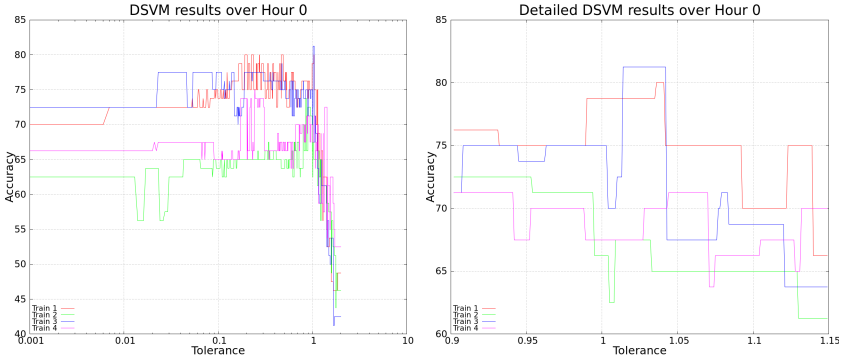
The classifier acts in different ways depending on ranges of tolerance. For example for tolerance above 1.05, the classifier does not change its accuracy but it remains at low accuracy (between 45% and 65% depending on the example). For this period, the success rate is too low and also the fluctuations depending on the training set chosen are huge. For tolerance values below 0.8 the accuracy also is stable, but with values between 62% and 67%, what means that in this range the classifier is more stable. Between values of 0.8 and 1.05 there is a raise of accuracy for the lower tolerance values having it a maximum at tolerance 0.97. After this value, it drops dramatically until it stabilizes at 1.05. All the examples follow this pattern with less than five points between them, what makes this period a stable one.

The best tolerance that can be inserted in the ARSVM classifier for detecting Hour 0 events in period 3 is 0.97 having a success rate of 65% to 73%. The tolerance range of this maximum accuracy is very short, dropping the success

rate when moving tolerance from this value. It would be interesting to develop more tests when a drift is present in the noses and with training sets with more patterns.

Dynamic SVM

In figure 3.40 two graphs represent the accuracy values of DSVM classifier using training and test sets with Hour 0 event patterns recorded during period 3. Graph 3.40a has the X axis in a logarithmic scale while graph 3.40b is in arithmetic scale. The tolerance range goes from 0 to 2.



(a) Accuracy of DSVM classifier. (b) Detailed view.

Figure 3.40: DSVM classifier accuracy as a function of tolerance variation over the time hour 00:00-1:00 against the rest of hours. There is a high accuracy variability between examples.

The best results can be seen in the zoom in graph 3.40b having an accuracy between 72% and 77% for tolerance values lower than 1.

For these graphs there are not radical changes in the behavior of the accuracy depending on the tolerance value, but is obvious that, from tolerance 0 to 1, there is an ascending trend in all the examples with around 10 points of accuracy between different trained classifiers and having a variability of around 5 points using the same training set for small tolerance variations. There is a maximum around 1 and then the accuracy drops to values near 50%, demonstrating that the classifier is as useful as the null classifier for tolerance values above 2.

So we can conclude that the best accuracy for DSVM classifier for detecting Hour 0 events in period 3 comes when tolerance has a value of 0.9 having success rates of 72% and 77%.

Classifying conclusions for reactor unload event detection

After analyzing the results for Hour 0 detection over the four classifiers we have these results: SVM classifies with accuracy between 60% and 75% while CSVM drops it to 58% to 70%. ARSVM has a hit rate between 65% and 73% and DSVM gives results from 72% to 77%. The best classifier for hour 0 detection is DSVM with a tolerance of 0.9.

We must be concerned to see that CSVM achieves values below SVM. Usually, as CSVM being only SVM with cross validation introduction, the results should improve or, in the worst case, be the same. But this does not happen for Hour 0 events. A possible explanation could be that there are not enough patterns for extracting the features of the events and that makes CSVM reach lower values.

Chapter 4

Conclusions and discussion

A low-cost nose sensor network has been successfully installed and used for monitoring a biodiesel plant for two years. The network is composed of several noses located in different places to detect events through the whole factory. The results presented in this work seem to indicate that is possible to detect security events, plant activity and production. The system presented in this document is modular and there was no problem in combining already installed artificial noses with new-in-the-system ones. The first system model installed in the factory had only one nose that could record time series during six months, but then the sensor collapsed. It was possible to change the nose without modifying the rest of the system, so introducing or changing noses is an easy task in this defined system. The communication of the noses with the central system, a computer located in the factory, has never broke because of internal problems. Sometimes there were outages, but rebooting the central computer reestablished the system. So we can conclude that the system is stable and not external error prone. The acquisition program has been modified, making different instances to record the sensor values each second. This program has been recording correctly for more than six consecutive months. We can conclude that the data acquisition is stable enough to not lose data during long periods. So with all the acquisition system installed and working correctly, it has been demonstrated that this system has te capability to detect security related events and also to characterize plant's activity with the advantages an artificial nose gives, such as low-cost or making the system not invasive.

A set of time series classification experiments based in SVM classifiers (SVM CSVM ARSVM and DSVM) have been developed for five different event types using two kinds of time series sets, one with data from two sensors and other with data from fifteen. Different variables, depending on the classifier, have been sweep to detect their best values: gamma, cost and tolerance for SVM and only tolerance for CSVM ARSVM and DSVM. Different training sets are used for each classifier, event and type of time series set. For events recorded by fifteen sensors only SVM and CSVM has been used. Next conclusions were given:

- Caustic soda load events: The best classifier for fifteen sensor values examples has been SVM with a success rate between and 70% 80% with tolerance 0.9 cost 10 and gamma 1. For two sensor values examples DSVM classifier gives an accuracy between 75% and 85% with tolerance value of 0.95.

- Methanol load events: For two sensors, the best classifier has been CSVM with a tolerance value of 0.1 and a success rate from 73% to 83%. For fifteen, the best classifier is SVM with tolerance 0.9, cost 10 and gamma 1 giving an accuracy going from 65% to 80%.

- Reactor startup events: Only being tested with SVM and CSVM over period three, the best classification is given by CSVM classifier with a tolerance of 0.5 and a success rate going from 97% to near %100

- Reactor unload events: These events have also only been tested with fifteen sensors and for SVM and CSVM classifiers, being CSVM with a tolerance value of 0.5 the best classifier giving an accuracy between 97% and almost 100%.

- Hour detection: Only applied to discriminate time series recorded from 00:00 to 00:59 against the rest of hours with fifteen sensors, the best results are given by DSVM classifier with a tolerance of 0.9 giving an accuracy between 72% and 77%

Artificial noses have noise, drift and history dependent sensitivity. Nose reaction to different events is subordinated in some level to the previous history of the signals and also to the quantity of material loaded or unloaded, being different for each event. So the defined environment is complex for classification tasks, however for almost all the events the classifiers reach a 75% of detection. The experiments have demonstrated that these events have some hidden features that can be extracted by the SVM to automatically classify them. More experiments for each event can be developed to increase the success rate. Introducing new variables to the classifier such as the quantity of material loaded or the previous history of the sensors can also raise the hit rate. The available time of this investigation have set a limitation for the number of experiments, but the bases for future research have been established. The event time stamps have been manually annotated by the workers in the plant, and the time accuracy is not assured. Timestamps could be checked and adjusted from the time series and improved Protocols could be proposed to improve timestamps accuracy. In this work is shown that a low-cost nose sensor network is not only useful for monitoring leak events but can also be used for the characterization of the plant activity during long periods. This arises from the fact that the factory routines (rest or working periods, load events, productivity, etc.) are reflected in the acquired signals.

Not only SVM based classifiers can be used for classifying the events, also other options are available, such as random forests, logistic regression, or artificial neural networks. Being all of them (including SVM based ones) automatic event classifiers, they can give to the system new functionalities focused on the automation of the plant, like performing parametric adjustments depending on the odor detected or automatizing security related actions such as access or ventilation system control.

To improve the classification ability of the classifiers, different future research, meeting low-cost requirements, can be done: It seems that for reactor events, the noses in reactor area are the ones that provide more information, so installing more low-cost noses in this area can give better results. The same happens with soda an methanol load events, maybe introducing more noses in the load room increase the accuracy of the classifiers. So the system can easily grow in number of noses. Feeding classifier learning process with more event examples or with multivariate information from multiple noses can make them to extract more features and increase the success rate. Also for reactor events, ARSVM and DSVM should be checked. For generating the examples for the classifiers is possible to change the time series sampling rate and introduce different combinations of all the possible time series. Reactor startup and unload define starting and ending activity periods, so the system can be used to classify the productivity of the plant. Is interesting to investigate the drift suffered by the sensors and also changes in output that suffer the system when replacing a nose. Adaptive systems can be useful for this drift and sensor replacement and can optimize the characterization, detection and classification capabilities[Herrero-Carrón et al., 2014]. Other capability that has not been in-

tegrated in the system is the option to have mobile artificial noses [García-Saura et al., 2014], but this feature could improve the system efficiency.

You can use different collaborative capabilities of the nose sensor network. If we introduce a real time analyzer, for making it feasible, it can analyze continuously only one nose and include other signals in punctual analysis, when the data given by the single nose is not enough to give a clear analysis. If integrated, modulation over a sensor can be defined by other sensor values.

The system designed can be useful for different purposes. The most important one is for security reasons. It can be used to detect leaks in different kind of factories or in laboratories. Also can be used for analyzing air quality control. For example it can be installed in other type of factories or can be used for controlling the effect of the air quality in productivity, like in a classroom with students to analyze if the air quality affects their marks or in a any kind shop to check if the air characteristics affect the sales. It can also be used to quantify productivity of different environments, such as a factory like Elda biodiesel plant or other purpose factories, but it can also be used to analyze the use that is being given to a classroom or a conference room or even the productivity of a bakehouse.

Appendix A

OlusElda program

OlusElda program was created by David Yañez for capturing the nose sensor data and saving it in a physical support and also for monitoring the process through a GUI. When installing the noses in the Elda plant, some problems arose and the program was forced to be modified. Next are the changes introduced in the program:

- Created three instances of the program, one for each COM: Each nose has an identifier, so for each program, the noses not present on that COM were disabled having an scheme similar to the following:

  Code A.1: Example of nose identifiers enabled for each com

```
tramatxrx(":0160503FF", tramar, 10, 4); //COM4_TGS2600
tramatxrx(":0360503FF", tramar, 10, 4); //COM4_TGS2602
//tramatxrx(":0560503FF", tramar, 10, 4); //COMX_ Not used
//tramatxrx(":0760503FF", tramar, 10, 4); //COM5_TGS2611
tramatxrx(":0960503FF", tramar, 10, 4); //COM4_TGS2611
//tramatxrx(":1160503FF", tramar, 10, 4); //COM5_TGS2620
```

  The filter for reading noses was implemented with a simple if when asking for the data registered in the noses. This way only the noses present in the selected COM are asked for the sensor signal:

  Code A.2: Selection of the noses to read from

```
if(cont2==1 || cont2==2 || cont2==5) //COM4
//if(cont2==4 || cont2==6)          //COM5
//if(cont2==3)                      //COM3
```

  Each different instance of the program has an if activated for reading different noses depending on the instance of the program.

- Capturing the humidity and radiation values:

  All noses have one analog channel linked to an address. The next code is added at the header of the file.

  Code A.3: Humidity and radiation addresses

```
#define LH1     ":023010000"        //Read level analog1.
```

```
#define LH2     ":043010000"          //Read level analog2.
#define LH3     ":063010000"          //Read level analog3.
#define LH4     ":083010000"          //Read level analog4.
#define LH5     ":103010000"          //Read level analog5.
#define LH6     ":123010000"          //Read level analog6.

#define LPG     1                     //Number of lectures to save one value.

unsigned int hum[8];                  //Temperature value

char *th[8]={LH1,LH2,LH3,LH4,LH5,LH6}; //Frames to send analogical read.
```

Each of the addresses for each analog channel are defined to access that register. Also the original program read ten values before saving one making ten readings each second, this is changed to one reading to make the system faster. A new global variable called hum is created to save the analog channel values in files. Other variable needed is *th*, that consists in six strings to save the data received from each nose's analog address.

The next code is included after reading temperature channel for each nose.

Code A.4: Humidity and radiation values capturing

```
/*Done for each nose*/
for(cont2=0;cont2<6;cont2++)      //Goes through six noses.
{

   /*Initialize variables where to save the data*/
   for(cont=0;cont<10;cont++)
      tramar[cont]=0;

   /*Read the analog values*/
   tramatxrx(th[cont2], tramar, 10, 4);

   /*Saves only the value from the data received*/
   for(cont=0;cont<4;cont++)
      dato2[cont]=tramar[cont+6];

   /*Prints analog value in the GUI*/
   SetWindowText(hwndEdHum[cont2], dato2);

   /*If data captured is meaningful we save it*/
   previo=ascii2int(dato2);
   if(previo>=0)
      hum[cont2]=previo;
}
```

All analog channels are read and then, when transforming the data, is when the meaningless columns are deleted. That way a new analog sensor can be implemented without modifying the code.

- Introduction of a new column for each nose in the output file for saving the analog channel values.

Code A.5: Save the analog channel's values in the file

```
if(puntos%LPG==0)
{
   fprintf(fdatos,"%d\t",nivel[cont2]);
   fprintf(fdatos,"%d\t",termo[cont2]);
   fprintf(fdatos,"%d\t",hum[cont2]); //Line introduced
}
```

Only the fifth code line is introduced to print the analog channel value after the temperature and the sensor values.

- Change the filename to distinguish each instance's output:

Code A.6: Change in the output's filename

```
char fech[34] = "COM3 \0";

tiempo=7200+time(NULL);
fecha=asctime(gmtime(&tiempo));
for(cont=0;cont<24;cont++)
{
   fech[cont+5]=fecha[cont];
}

fech[18]='_';
fech[21]='_';
fech[29]='.';
fech[30]='x';
fech[31]='l';
fech[32]='s';
fech[33]=0;
fdatos = fopen(fech, "w");
```

When the program is saving the data to a file, if it is creating it, now adds at the beginning the COM's name (COM3, COM4 or COM5 depending on the program instance), and then adds the date just as *gmtime()* function returns.

- Include Time stamp and analog boxes in GUI:

Code A.7: Print GUI labels

```
case WM_PAINT:
{
   hdc=BeginPaint (hwnd, &ps);
   TextOut (hdc, 476, 85, "Hum ", 4);
   TextOut (hdc,  5, 82, "Day" , 3);
   TextOut (hdc, 55, 82, "Hour", 4);
   TextOut (hdc, 108, 82, "Min" , 3);
   TextOut (hdc, 154, 82, "Sec" , 3);
   TextOut (hdc, 200, 82, "." , 1);
}
```

This code fragment prints the fixed labels in the GUI for knowing what values are in each box. It is only executed when starting the executable.

Code A.8: Generate boxes for printing time and analog values

```
case WM_CREATE:              //Create boxes.
{
   hwndEdDay=CreateWindow("edit",NULL,      //Edition "Day".
   WS_BORDER | WS_CHILD | WS_VISIBLE,
   32,80,20,20,hwnd,(HMENU) 5,((LPCREATESTRUCT)lParam)->hInstance ,NULL);

   hwndEdHour=CreateWindow("edit",NULL,      //Edition "Hour".
   WS_BORDER | WS_CHILD | WS_VISIBLE,
   87,80,20,20,hwnd,(HMENU) 5,((LPCREATESTRUCT)lParam)->hInstance ,NULL);

   hwndEdMin=CreateWindow("edit",NULL,      //Edition "Min".
   WS_BORDER | WS_CHILD | WS_VISIBLE,
   133,80,20,20,hwnd,(HMENU) 5,((LPCREATESTRUCT)lParam)->hInstance ,NULL);

   hwndEdSec=CreateWindow("edit",NULL,      //Edition "Sec".
```

```
16     WS_BORDER | WS_CHILD | WS_VISIBLE,
17     179,80,20,20,hwnd,(HMENU) 5,((LPCREATESTRUCT)lParam)->hInstance ,NULL);
18
19     hwndEdMillisec=CreateWindow("edit",NULL,      //Edition "Millisec".
20     WS_BORDER | WS_CHILD | WS_VISIBLE,
21     205,80,30,20,hwnd,(HMENU) 5,((LPCREATESTRUCT)lParam)->hInstance ,NULL);
22
23     for(cont=0;cont<6;cont++)
24     {
25        hwndEdHum[cont]=CreateWindow("edit",NULL,//Edition "Analog channel".
26        WS_BORDER | WS_CHILD | WS_VISIBLE,
27        240+cont*40,80,35,20,hwnd,(HMENU) 12,
28        ((LPCREATESTRUCT)lParam)->hInstance ,NULL);
29     }
30  }
```

The code generates five boxes for printing day, hour, minute, second and millisecond of the last recording. Also creates six boxes under temperature and sensor boxes for printing the last recorded values of the analog sensors.

- Introduction of a column with the capture time in the output file and print the time with milliseconds in the GUI:

Code A.9: Time stamp handlers and variables for representation in the GUI

```
1  static HWND hwndEdDay;              //Day edition handler.
2  static HWND hwndEdHour;             //Hour edition handler.
3  static HWND hwndEdMin;              //Minute edition handler.
4  static HWND hwndEdSec;              //Second edition handler.
5  static HWND hwndEdMillisec;         //Millisecond edition handler.
6  static HWND hwndEdHum[6];           //Analog channel edition handler.
7
8  SYSTEMTIME st;
9  char timeBuff[4];
```

New handlers are added to print date values in the GUI. Also *st* variable is created for reading the date from the system. *timeBuff* is a temporal variable to print the values into the GUI.

Code A.10: Time stamp record in output file and in GUI

```
1  if(puntos%LPG==0)
2  {
3     GetSystemTime(&st);
4     fprintf(fdatos,"Y%04dM%02dD%02dH%02d:%02d:%02d.%03d\n", st.wYear,
5        st.wMonth, st.wDay, st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
6     /*Print the Day*/
7     sprintf(timeBuff, "%02d", st.wDay);
8     SetWindowText(hwndEdDay, timeBuff);
9     /*Print the Hour*/
10    sprintf(timeBuff, "%02d", st.wHour);
11    SetWindowText(hwndEdHour, timeBuff);
12    /*Print the Minutes*/
13    sprintf(timeBuff, "%02d", st.wMinute);
14    SetWindowText(hwndEdMin, timeBuff);
15    /*Print the Seconds*/
16    sprintf(timeBuff, "%02d", st.wSecond);
17    SetWindowText(hwndEdSec, timeBuff);
18    /*Printf the Milliseconds*/
19    sprintf(timeBuff, "%03d", st.wMilliseconds);
20    SetWindowText(hwndEdMillisec, timeBuff);
21 }
```

This code is introduced after reading all the nose addresses, so the time is placed at the end of each line. Also when each time reading is done,

the time represented in the GUI is also updated through the function *SetWindowText()*.

These changes made possible to have a clean register of all the data needed for making the experiments.

Appendix B

Artificial nose housing design and printing

As told in chapter 2, a housing for the hardware has been designed and printed for protecting the nose to the environmental hazards making it to work properly for longer times (see Figure 2.11).

B.1 Housing design

Openscad has been used for designing the pieces and transforming them into STL format [Openscad page, 2014].

The housing consists in top and a bottom parts having the next code:

Code B.1: Bottom shield scad

```
difference(){
        //Cubo de fuera
        cube(size=[48,48,6], center = true);
        //Cubo de dentro
        translate([0,0,2]) cube(size=[43,43,6], center = true);
        // Cubo Usb
        translate([0,12.5,-1]) cube(size=[14,13,6], center = true);
        //Cubo pines
        translate([-6.5,-16.5,-3]) cube(size=[27,6,6], center = true);
        //Cubo luz
        translate([-15,-4.5,0.5]) cube(size=[3,6,7.7], center = true);
        //Nariz
        translate([18,-16,4]) rotate([90,0,90]) cylinder(20,5,5);

}
translate([12,22.75,4]) cube(size=[4,2.5,5], center = true);
translate([-12,22.75,4]) cube(size=[4,2.5,5], center = true);
translate([12,-22.75,4]) cube(size=[4,2.5,5], center = true);
translate([-12,-22.75,4]) cube(size=[4,2.5,5], center = true);
translate([10,-13,-3]) rotate([0,0,0]) cylinder(10,1.7,1.7);
```

Code B.2: Top shield scad

```
difference(){
        cube(size=[48,48,9], center = true);
        translate([0,0,2]) cube(size=[43,43,9], center = true);
        translate([16,-20.5,3]) rotate([90,0,0]) cylinder(20,5,5);
        translate([22.5,12,3]) cube(size=[5,5,5], center = true);
```

```
6           translate([22.5,-12,3]) cube(size=[5,5,5], center = true);
7           translate([-22.5,12,3]) cube(size=[5,5,5], center = true);
8           translate([-22.5,-12,3]) cube(size=[5,5,5], center = true);
9   }
10          translate([-14,-3,0]) cube(size=[20,3,9], center = true);
11          translate([4,15,0]) cube(size=[15,3,9], center = true);
12          translate([-4,6,0]) cube(size=[3,21,9], center = true);
13          translate([12,11,0]) cube(size=[3,11,9], center = true);
```

After having the design, its construction is necessary. For this purpose, a 3D printer was available for printing it.

B.2 3D printer

A 3D printer is a machine that allows 3D printing. 3D printing is the process of making a three-dimensional object of almost any shape from a 3D digital model or other digital data source through additive processes in which successive layers of material are laid down under computer control. There are different commercial and personal 3D printers that allow this kind of printing.

For creating the housing for the noses in Elda plant and also for creating the car chassis for the robots where all the sensors and electronics will be placed, a 3D printer has been used. The 3D printer was granted by Carlos García, being this a *RepRap* 3D printer (figure B.1) [Reprap page, 2012b].

RepRap self-replicates its own parts allowing to change the structure of the own printer or replicating it creating another one. There exist several models of *RepRap* printers, having chosen the simplest model for its low cost and versatility, the *Printbot* [Reprap page, 2012a]. Not the whole printer is made of plastic parts created by other *Printbot*, you need to get an extruder, a fan, chassis and engines, the controller and also the bed on which the printing process is done. This printer can make small plastic pieces in half an hour and bigger ones augmenting the time it requires. The pieces can reach sizes of 20 x 20 x 15 cm.

The material used for printing the figures is thermosoftening plastic because it becomes pliable or moldable above a specific temperature and returns to a solid state upon cooling. Printing with thermosoftening plastic consists on a plastic wire of a given diameter that passes through a hole that heats it, called extruder.

The most used thermoplastics are ABS or PLA. The prices of both materials are similar but have different characteristics:

- ABS: The melting point of this material is 230°C, considered high in thermoplastics. The material sticks hardly to surfaces, this is why is mandatory to have a printer with heat bed. On reaching its melting point, ABS emits gases which in high concentrations can be harmful, but is not dangerous at low concentrations, so a single printer in a room has no risk. The ABS can be easily treated after printing, and the finish is still good. It is a tough material having some flexibility.

- PLA: The melting point is lower than te ABS, 185°C. It does not emit dangerous gases and has a more wide color range because of the material with which it is made (is obtained from corn starch making it biodegradable). It does not need a heat bed because of its melting point, but at low

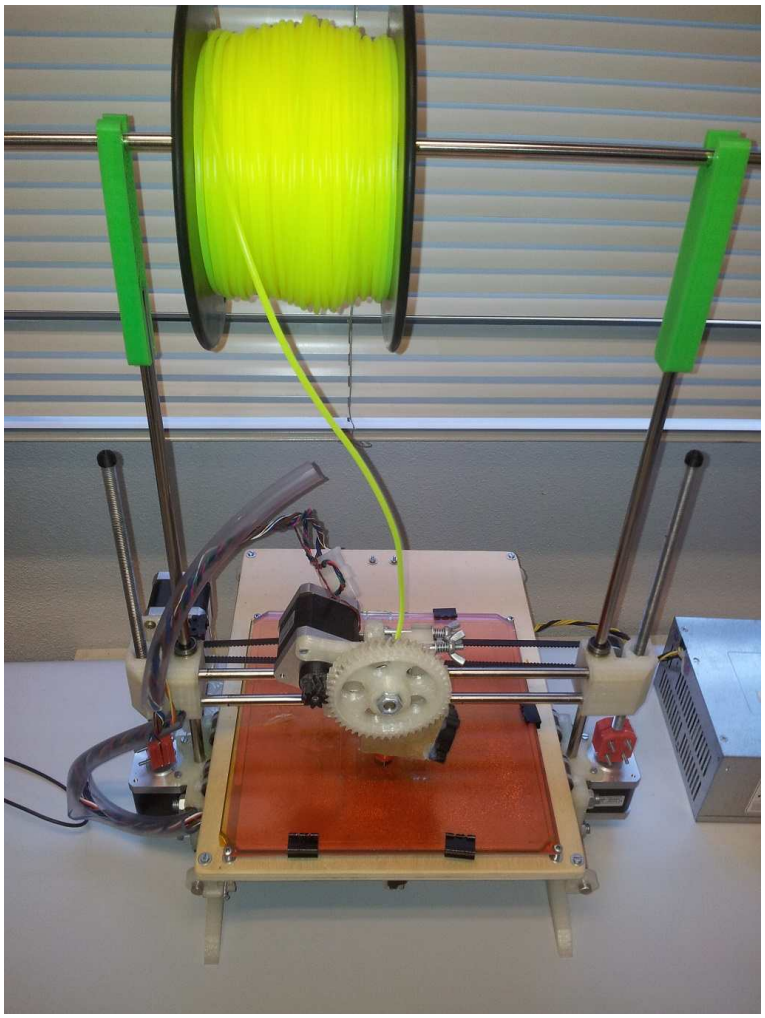


Figure B.1: fig: General view of the 3D printer.

temperatures such as 60ºc in continuous exposure it starts to decompose and break up. The treating of the figures after printed, such as filing, pasting or piercing, is harder than ABS. Also this material is very rigid.

We have chosen PLA thermoplastic to print with because of being cheaper in energy and being less dangerous because of not emitting noxious gases at its melting point. Also having more available different colors is an advantage to the project, because many robots will be made with it and having them in different colors is a good way to differentiate them easily.

Sometimes if the base of the printed figure moves because of not being attached to the bed the print fails. This usually happens printing just over the bed. One approach to solve this is to heat up the bed. In this printer the print bed can be self heated up to 100 ºC to make the printed plastic piece affixed to the base. It generates too much heat and requires also much energy, so other

approach is used: before starting to print, a layer of hair spray is placed down on the bed to attach the first layer of the plastic to the base.

To move the engines, the controller has not enough power, so an extra power supply is necessary. in this case, a desktop computer power supply is integrated in the system.

B.3 *Cura 13*

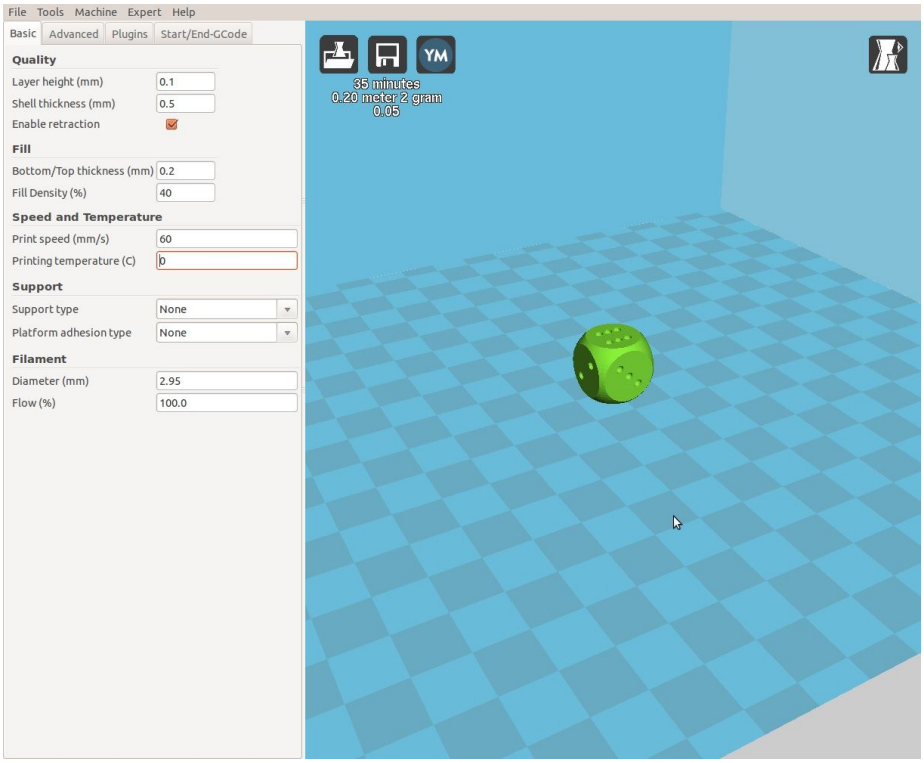


Figure B.2: *Cura 13* GUI.

For connecting the 3D printer with the computer, a program is needed. Different free licensed programs exist for this purpose, but the one chosen has been *Cura 13* [Cura page, 2014] because of being easy to configure and to use (Figure B.2). Cura 13 not only offers a friendly GUI but also allows modifications in the figures such as increase the size, rotate or invert it. Also transforms the STL format (STL or Standard Tessellation Language is a file format for representing 3D figures) into Gcode format (processed by a piece of software called a "slicer" which converts the model into a series of thin layers and instructions to move the extruder of the printer). An advantage of this program is that detects automatically the port where the USB cable is connected. A factor to consider is that this program requires superuser rights. Other useful program with less bugs is the *Printrun* [Printrun page, 2014] which works as a GUI but has not a *slicer* incorporated, so its use has been reduced for debugging

purposes only.

B.3.1 Configuring *Cura 13*

Cura 13 allow us to modify many parameters of the printer. As it is home made, parameters have had to be calculated, although most of them have been estimated by Carlos García. Different images have been taken with these parameters and the most important are discussed.

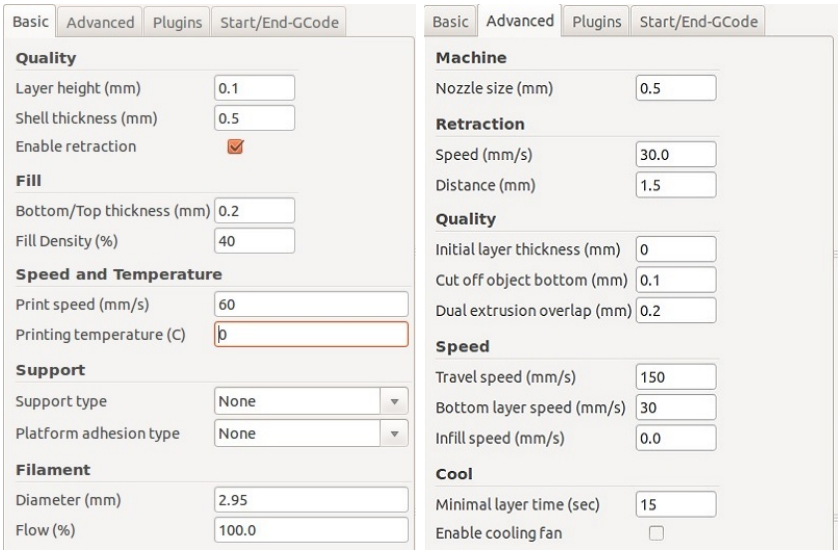


Figure B.3: *Cura 13* basic preferences.

Figure B.4: *Cura 13* advanced preferences.

- Basic preferences (Figure B.3): The most important preferences for performing the printing depending on the figures are listed here.
  - Layer height defines the detail with which the figure is printed, values can go from 0.1 (for a detailed but slow printing) to 0.25 (for a cruder but faster figure printing).
  - The shell thickness specifies the width of the walls of the figures. 0.5 is a good value unless there are thin walls, then the value can be reduced up to 0.2.
  - Bottom/Top thickness has the same effect as Shell thickness but for the upper and lower walls. This value should be a multiple of the Layer height, being the double of it a good value.
  - Fill density defines the quantity of material that the figure will be filled with, a 100% can turn into a distortion of the resulting figure and low values can make the figure to collapse. Normal values are from 20 % to 40 %.
  - Support type can be activated for figures that have hanging parts. This option makes a support for building those parts that, after the impression, have to be removed.

- Advanced preferences (Figure B.4): The values in this tab are better to not be changed because they are intrinsic values of the printer.

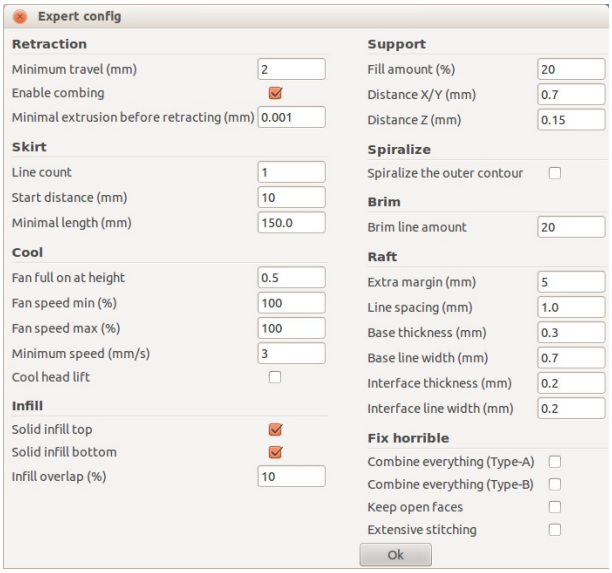


Figure B.5: *Cura 13* expert settings.

- Expert preferences (Figure B.5): This tab is located in the menu *expert* clicking in *open expert settings* on the top bar. These preferences are better to not change because they can make the printer not work properly, so is better to put the values exactly as in the image.

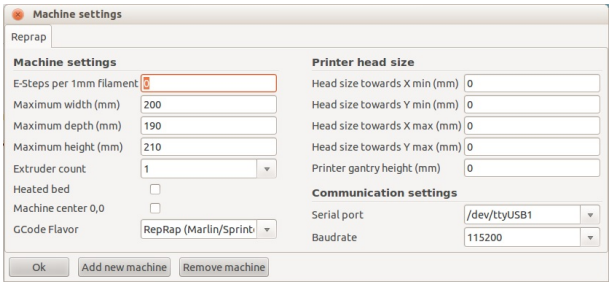


Figure B.6: *Cura 13* machine settings.

- Machine preferences (Figure B.6): Located in the menu *file* clicking in *machine settings*, these preferences are also better to be put as in the image to make the printer work properly. The only setting that can be modified without any risk is the serial port, just defining the one where the printer is or just let the program to automatically detect it.
- Start G-code: These lines specify the behavior of the printer just before printing to prepare it. Next lines should be written without modifications

in the *start.gcode* tab inside *Start/End-GCode* tab in the main window of the GUI.

Code B.3: Start G-code

```
;Sliced {filename} at: {day} {date} {time}
;Basic settings: Layer height: {layer_height} Walls: {wall_thickness} Fill:
     {fill_density}
;Print time: {print_time}
;Filament used: {filament_amount}m {filament_weight}g
;Filament cost: {filament_cost}
G21       ;metric values
G90       ;absolute positioning
M107      ;start with the fan off
G28 X0 Y0 ;move X/Y to min endstops
G28 Z0    ;move Z to min endstops ; Disabled for Cupcake
G92 X0 Y0 Z0 E0       ;reset software position to front/left/z=0.0
G1 Z5.0 F50           ;move the platform down 15mm
G92 E0                ;zero the extruded length
G1 F200 E1            ;extrude 3mm of feed stock
G92 E0                ;zero the extruded length again
;go to the middle of the platform (disabled, as there is no need to go to the center)
;G1 X{machine_center_x} Y{machine_center_y} F{travel_speed}
G1 F{travel_speed}
```

- End G-code: Just as in the Start G-code tab, the best is to copy these instructions in the *end.gcode* tab inside *Start/End-GCode* tab in the main window.

Code B.4: End G-code

```
;End GCode
M104 S0                     ;extruder heater off
M140 S0                     ;heated bed heater off (if you have it)
G91                         ;relative positioning
G1 E-1 F300                 ;retract the filament a bit before lifting the nozzle, to
     release some of the pressure
G1 Z+0.5 E-5 F{travel_speed} ;move Z up a bit and retract filament even more
G90                         ;absolute positioning
G1 X10 Y180 F{travel_speed} ;move extruder to center-back, so the head is out of the
     way
M84                         ;steppers off
```

With these values correctly defined in the *Cura 13* program, the printer would be ready to start printing. The result of the printed housing can be seen in the Figure 2.11.

Appendix C

Autonomous robots

An initial study was done to include mobile collaborative noses to be able to move around the area of investigation in an autonomous way. For this purpose, the first raised idea was to use an autonomous mobile robot using a micro-controller.

C.1 SkyBot

The first option to use was the SkyBot robot, because there was already one mounted, it was cheap to replicate and it was modular. It was granted by Andrés Prieto-Moreno Torres.

C.1.1 Hardware

The basic robot has two boards:

- Skypic: Is the master board, which is in charge of administrating the inputs and outputs as to contain and execute the algorithm with which the robot works.
- Sky293: This board is in charge of controlling the servos and sensors. It communicates with Skypic board by connecting a pair of ports achieving the communication with a very trivial process [Skypic wiki page, 2007a].

The mobile robot has the two boards on top of it and two endless tracks as wheels powered by two modified servo motors to act with DC current. In front of the robot (at the left of the image C.3), a pair of bumpers are placed to make the robot know when it collides. Having two of them allows the robot to know which side the collision has happened.This easy mechanism allows to know a bit of the environment of the robot. Skypic board allows also to include four infrared sensors to make more complex environmental checks.

C.1.2 Configuration

For testing the robot, a software is needed for installing in Linux [Skybot wiki page, 2007e] or Windows [Skybot wiki page, 2007d]. Also is necessary to have a usb-serial cable to connect the robot to the computer.

Figure C.1: Skypic microcontroller board. Image extracted from the Skypic wikirobtics [Skypic wiki page, 2007b].

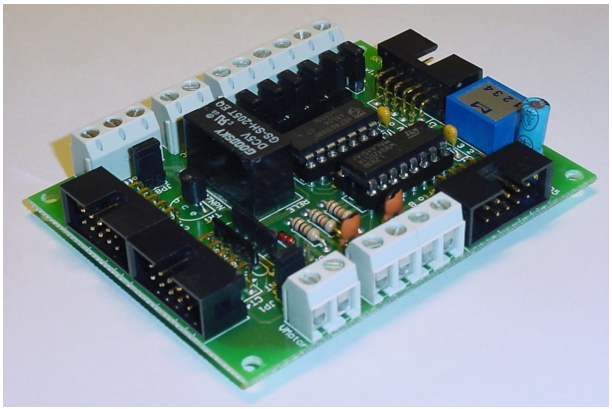


Figure C.2: Sky293 board. Image extracted from the Sky293 web page [Skypic wiki page, 2007b].

The most important software to use for testing the robot is:

- SkyBot-Test: This program allows to test in real time if the bumpers, infrared sensors and servos are working properly. The program shows an interface with buttons to send instructions to the robot and images to check the state of the different sensors [Skybot wiki page, 2007c]. If a sensor is not introduced in the system (for example a bumper or infrared sensor), the program works anyway. An image of the program is shown in image C.4
- SDCC: Is the compiller suite used for compilling the code generated in C [SDCC page, 2007]. The method to invoke the compiller by command line

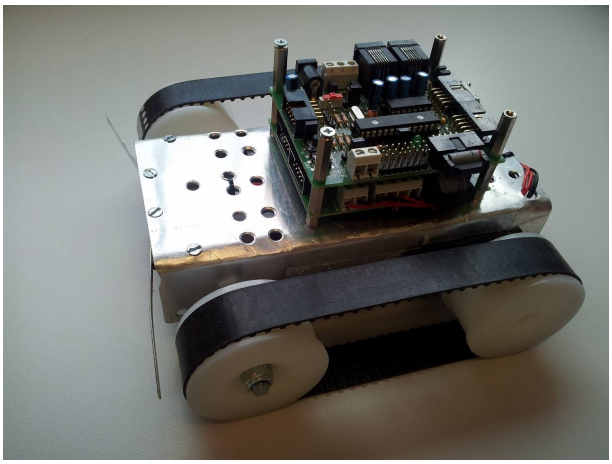


Figure C.3: Image of the mobile robot with the skypic and sky293 boards integrated.

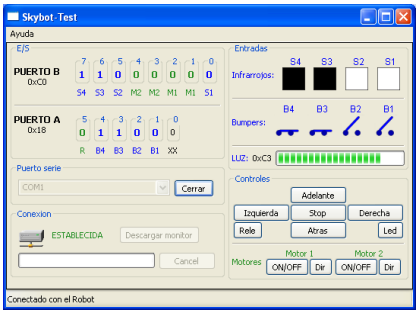


Figure C.4: SkyBot-Test program executed in windows [Skybot wiki page, 2007c].

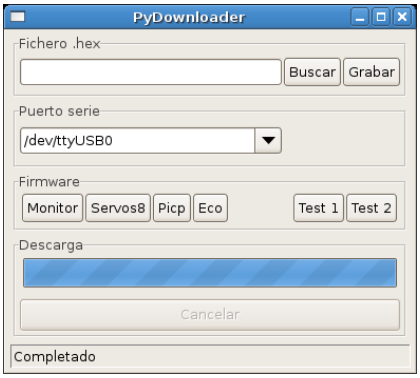


Figure C.5: Pydownloader executed in Linux [Skybot wiki page, 2007b].

in linux is the next:

```
1 sdcc -Wl-ainhx8m -mpic14 -p16f876a -o program program.c
```

A file called program.hex will be created. This file is the binary file of the program that must be downloaded via Pydownloader to the skypic board to be able to execute it.

- Pydownloader: Is a program that allows to download .hex files to the skybot in an easy way with a visual interface [Skybot wiki page, 2007b].For using this program, is essential to have a Bootloader installed in the microcontroller [Skybot wiki page, 2007a]. For loading files in the robot, is only needed to choose the .hex file and then click on the '*grabar* button. An image of the program is shown in image C.5

C.1.3 Software

Three programs were made during the testing of the robot.

Hello world

This program configures the port B of the robot to define which is input and output and then makes the robot run forward, backwards, to the left and to the right.

The file *pic16f876a.h* is a header file that provides memory access to the ports and methods to modify them.

The program uses execution time to decide when to change the status of the servos. A sleep function would be more precise and clean, but as being the purpose of this program so simple, it was not implemented.

Code C.1: hello_world.c

```
1  //-- Value to confuigure port B to work with skybot
2  //-- Bits RB0, RB5, RB6 y RB7 as inputs (sensors)
3  //-- Bits RB1, RB2, RB3 y RB4 as outputs
4  #define CONF_SKYBOT 0xE1 //-- Binary: 1110 0001
5
6  #define  ALANTE    0x1C
7  #define  ATRAS     0x16
8  #define  IZQUIERDA 0x1E
9  #define  DERECHA   0x14
10 #define  STOP      0x00
11 #define SENS_DER RB7
12 #define SENS_IZQ RB6
13 #define SKYBOT PORTB
14 #include <pic16f876a.h>
15
16 void main(void)
17 {
18        long i=0;
19        //-- configure port B
20        TRISB=CONF_SKYBOT;
21        //-- Stopped skybot servos
22        SKYBOT=STOP;
23
24        while(1){
25               if (i==300000){
26                      SKYBOT=ALANTE;
27               }
28               if (i==150000){
29                      SKYBOT=ATRAS;
30               }
31               if (i==450000){
32                      SKYBOT=IZQUIERDA;
33               }
34               if (i==600000){
35                      SKYBOT=DERECHA;
36                      i=0;
37               }
38        i++;
39        }
40 }
```

Easy Walk

This program does exactly the same as hello world code except for having interruptions. Also, using them, allow us to have the time controlled in an exact way.

Code C.2: timer0_delay.h

```
//-- Value to confuigure port B to work with skybot
//-- Bits RB0, RB5, RB6 y RB7 as inputs (sensors)
//-- Bits RB1, RB2, RB3 y RB4 as outputs
#define CONF_SKYBOT 0xE1 //-- Binary: 1110 0001

#define  ALANTE    0x1C
#define  ATRAS     0x16
#define  IZQUIERDA 0x1E
#define  DERECHA   0x14
#define  STOP      0x00
#define BUMPER_IZQ RA1
#define BUMPER_DER RA2
#define SKYBOT PORTB
#include <pic16f876a.h>
#include "timer0-delay.h"

void main(void)
{
       //-- configure port A as input
       TRISA = 0xFF;
       ADCON1=0x0E;
       //-- configure port B
       TRISB=CONF_SKYBOT;
       //--  Stopped skybot servos
       SKYBOT=STOP;

       //-- Configure timer to 0
       timer0_configurar();

  //-- Infinite loop
  while(1){
               //-- It's 50*10ms = 1 sec
               timer0_delay(50);
               SKYBOT=ALANTE;
               timer0_delay(50);
               SKYBOT=ATRAS;
               timer0_delay(50);
               SKYBOT=IZQUIERDA;
               timer0_delay(50);
               SKYBOT=DERECHA;
       }
}
```

The code $timer0\_delay.h$ is a header made to introduce timer delays of multiples of 10ms. It uses the header $pic16f876a.h$ to access to timer interruptions in function $timer0\_configurar()$. This function activates the timer and initializes the values needed.

$timer0\_wait(t0ini)$ defines the time $t0ini$ until the interruption $T0IF$ jumps, then waits until it changes its value.

When you call $timer0\_delay()$, it pauses $duration$*10ms the application by calling the function $timer0\_wait(t0ini)$ as many times as the number value defined in $duration$.

Code C.3: timer0_delay.h

```
#ifndef TIMER0_DELAY_H
#define TIME0_DELAY_H


#include <pic16f876a.h>


//-- Initial value of TMR0 to perform a 10ms pause max value 256
#define T_10ms 61

void timer0_wait(unsigned char t0ini)
```

```
12  {
13    //-- Initial value of timer
14    TMR0=t0ini;
15
16    //-- Interruption flag set to 0
17    T0IF=0;
18
19    //-- Wait until timer reaches 0
20    while(T0IF==0);
21  }
22
23  /*******************************************************/
24  /* Pause                                               */
25  /* INPUT : Pause length in hundredths (10ms)           */
26  /*******************************************************/
27  void timer0_delay(unsigned int duracion)
28  {
29    unsigned int i;
30
31    for (i=0; i<duracion; i++)
32      timer0_wait(T_10ms);
33  }
34
35  void timer0_configurar()
36  {
37    //-- Configure Timer to 0
38    //-- Timer mode
39    T0CS=0; PSA=0;
40
41    //-- Presscaler to 256
42    PS2=1; PS1=1; PS0=1;
43  }
44
45  #endif //-- TIME0_DELAY_H
```

Skirt objects

This program allows the robot to skirt obstacles. It uses the led, bumpers, interruptions, servos and buttons. So is a great program to use all the implemented characteristics in the robot. The main idea of the robot is, each time a bumper jumps, to turn a little in the opposite direction to where the object is.

This code initializes reset and push buttons, timer, bumpers, led and ports. Then it waits until we push the button to start the main algorithm. When the main loop starts, it begins checking continuously if a bumper or the button is pushed and, when the event is detected, it acts accordingly.

Take in account that this code does not allow to check two events that happen simultaneously, except for the two bumpers jumping at the same time, indicating that the robot is just in front of a wall.

Code C.4: skirt_objects.c

```
1   //-- Value to confuigure port B to work with skybot
2   //-- Bits RB0, RB5, RB6 y RB7 as inputs (sensors)
3   //-- Bits RB1, RB2, RB3 y RB4 as outputs
4   #define CONF_SKYBOT 0xE1 //-- Binary: 1110 0001
5
6   #define  ALANTE    0x1C
7   #define  ATRAS     0x16
8   #define  IZQUIERDA 0x1E
9   #define  DERECHA   0x14
10  #define  STOP      0x00
11
12  //-- Button and led of Skypic
13  #define PULSADOR RB0
14  #define LED    RB1
```

```
15
16  //-- Status for led and bumpers
17  #define ON 1
18  #define OFF 0
19
20  //-- Status of button
21  #define PULSADO  0
22  #define NO_PULSADO 1
23
24  //-- BUMPERS
25  #define BUMPER_IZQ RA2
26  #define BUMPER_DER RA1
27  #define SKYBOT PORTB
28  #include <pic16f876a.h>
29  #include "timer0-delay.h"
30
31  void main(void)
32  {
33          //-- configure port A as input
34          TRISA = 0xFF;
35          ADCON1=0x0E;
36          //-- configure port B
37          TRISB=CONF_SKYBOT;
38          //-- Stopped skybot servos
39          SKYBOT=STOP;
40          LED = PULSADOR;
41          //-- Configure timer to 0
42          timer0_configurar();
43          //-- Wait until button pushed
44          while (PULSADOR==NO_PULSADO);
45          timer0_delay(100);
46
47  //-- Main loop
48  while(1) {
49          if (PULSADOR==PULSADO){
50                  SKYBOT=STOP;
51                  timer0_delay(50);
52                  while (PULSADOR==NO_PULSADO);
53                  timer0_delay(50);
54          }
55          if (BUMPER_IZQ==ON && BUMPER_DER==OFF) {
56                  SKYBOT=ATRAS;
57                  timer0_delay(50);
58                  SKYBOT=DERECHA;
59                  timer0_delay(50);
60          }
61          if (BUMPER_IZQ==OFF && BUMPER_DER==ON) {
62                  SKYBOT=ATRAS;
63                  timer0_delay(50);
64                  SKYBOT=IZQUIERDA;
65                  timer0_delay(50);
66          }
67          if (BUMPER_IZQ==OFF && BUMPER_DER==OFF) {
68                  SKYBOT=ALANTE;
69          }
70          if (BUMPER_IZQ==ON && BUMPER_DER==ON) {
71                  SKYBOT=ATRAS;
72                  timer0_delay(50);
73                  SKYBOT=DERECHA;
74                  timer0_delay(50);
75     }
76    }
77  }
```

C.2 Arduino board

The idea was to use threads and multitasking to work with the robot, instead of an active wait. That way, each time a bumper triggers, the nose, communicated with the robot, would know where the obstacle is, but the PIC does not allow

to launch an interruption each time port B changes, and in this port is where the bumper status is located.

On other hand, the robot was not mechanically easy to replicate, because it has hand made pieces. The chassis has to be made in a non serial way.

Other options were taken in account, reaching the resolution to change the SkyBot for an Arduino board. The Arduino micro-controller allows to use shields to integrate new functionalities, such as wifi, xbee, a integration board for sensors, cable net interfaces, monitor output, etc. [Arduino page, 2013c].

There are many different Arduino boards, but the ones we chose were Arduino Leonardo [Arduino page, 2013a] and Arduino Mega 2560 [Arduino page, 2013b].

- Arduino Mega 2560: This board was chosen because of its versatility with shields and micro-controller outputs and inputs (having in total 54 digital and 16 analogical i/o). Other great advantage of this board is that it has 256 kBytes of flash memory, that allows "big" programs to be loaded , so the memory will not be a problem. This board allow to create a small sized robot without losing capabilities or sensor variety.

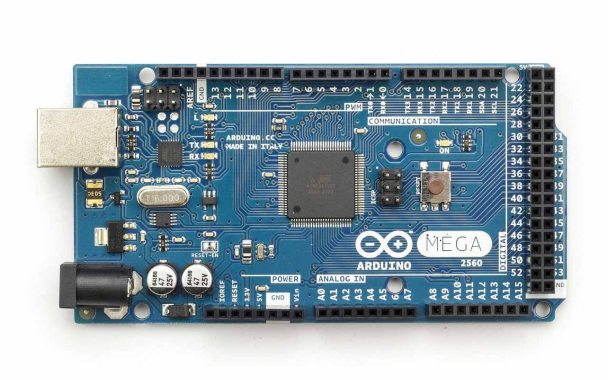


Figure C.6: Image of the Arduino Mega 2560 board.

- Arduino Leonardo: On the other side, we have also focused in a small board. Is smaller and easy to carry, ideal for having it in the pocket of a worker or located in small fixed or mobile objects. The Arduino Leonardo can allow the project to have more mobile noses apart of the ones in the robots. Having only 20 digital and 12 analogical i/o and a memory of 32 kBytes, this board allow less sensors, but it's worth because of its size.

Because of the amount of work that this part of the project was generating, the robot design and mount with all its peripherals is being carried out by Carlos García Saura in his bachelor thesis. Resuming the work from this point, Carlos García has gone on with the robot having the results posted in the project's web page [GNBot page, 2013].

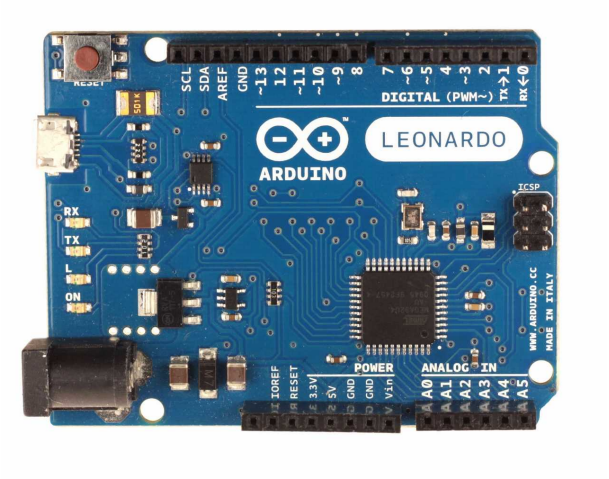


Figure C.7: Image of the Arduino Leonardo board.

Bibliography

H. Alause, F. Grasdepot, J.P. Malzac, W. Knap, and J. Hermann. Micromachined optical tunable filter for domestic gas sensors. *Sensors and Actuators B: Chemical*, 43(1-3):18–23, September 1997. ISSN 09254005. doi: 10.1016/S0925-4005(97)00139-1. URL `http://www.sciencedirect.com/science/article/pii/S0925400597001391`.

C. Alippi and M. Roveri. Just-in-Time Adaptive Classifiers—Part I: Detecting Nonstationary Changes. *IEEE Transactions on Neural Networks*, 19(7):1145–1153, July 2008a. ISSN 1045-9227. doi: 10.1109/TNN.2008.2000082. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4470009`.

C. Alippi and M. Roveri. Just-in-time adaptive classifiers-part II: designing the classifier. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 19(12):2053–64, December 2008b. ISSN 1941-0093. doi: 10.1109/TNN.2008.2003998. URL `http://www.ncbi.nlm.nih.gov/pubmed/19054730`.

Arduino page. Web page of Arduino Leonardo. `http://arduino.cc/en/Main/ArduinoBoardLeonardo`, 2013a.

Arduino page. Web page of Arduino Mega 2560. `http://arduino.cc/en/Main/ArduinoBoardMega2560`, 2013b.

Arduino page. Web page of Arduino Shields. `http://playground.arduino.cc/Main/SimilarBoards#goShie`, 2013c.

K. Arshak, E. Moore, G.M. Lyons, J. Harris, and S. Clifford. A review of gas sensors employed in electronic nose applications. *Sensor Review*, 24(2):181–198, 2004. ISSN 0260-2288. doi: 10.1108/02602280410525977. URL `http://www.emeraldinsight.com/10.1108/02602280410525977`.

N.E. Barbri, E. Llobet, N.E. Bari, X. Correig, and B. Bouchikhi. Electronic Nose Based on Metal Oxide Semiconductor Sensors as an Alternative Technique for the Spoilage Classification of Red Meat. *Sensors*, 8:142–156, 2008.

S. Bermejo. Independent component analysis for solid-state chemical sensor arrays. *Applied Intelligence*, 24(1):61–73, February 2006. ISSN 0924-669X. doi: 10.1007/s10489-006-6930-3. URL `http://dx.doi.org/10.1007/s10489-006-6930-3`.

A.Z. Berna, J. Lammertyn, S. Saevels, C.D. Natale, and B.M. Nicolai. Electronic nose systems to study shelf life and cultivar effect on tomato aroma profile. *Sensors and Actuators B: Chemical*, 97(2-3):324–333, February 2004. ISSN 09254005. doi: 10.1016/j.snb.2003.09.020. URL `http://www.sciencedirect.com/science/article/pii/S0925400503007305`.

J. Brezmes, E. Llobet, X. Vilanova, J. Orts, G. Saiz, and X. Correig. Correlation between electronic nose signals and fruit quality indicators on shelf-life measurements with pinklady apples. *Sensors and Actuators B: Chemical*, 80 (1):41–50, November 2001. ISSN 09254005. doi: 10.1016/S0925-4005(01) 00867-X. URL `http://www.sciencedirect.com/science/article/pii/S092540050100867X`.

T.M. Buck, F.G. Allen, and Dalton M. Detection of chemical species by surface effects on metals and semiconductors. In Surface effects in detection. *Bregmand, J.I., Dravnieks, A., Eds.; Spartan Books Inc.*, pages 1–27, 1965.

W.J. Buttner, G.J. Maclay, and J.R. Stetter. An integrated amperometric microsensor. *Sensors and Actuators B: Chemical*, 1(1-6):303–307, January 1990. ISSN 09254005. doi: 10.1016/0925-4005(90)80220-T. URL `http://www.sciencedirect.com/science/article/pii/092540059080220T`.

J. Cai and D.C. Levy. Source Direction Detection based on Stationary Electronic Nose System. *Journal of Applied Mathematics*, pages 179–183, 2007.

G.A. Carpenter and S. Grossberg. Adaptive Resonance Theory. *Encyclopedia of Machine Learning*, 104:19–26, 2009.

H. Chen, R.A. Goubran, and T. Mussivand. Improving the Classification Accuracy in Electronic Noses Using Multi-Dimensional Combining (MDC). *Analysis*, 2(Mdc):587–590, 2004.

P. Chomtip. Chemical Substance Classification By Electronic Noses. *Science, Computer and Rama, Science Medical Care*, pages 68–72, 2009.

M.P. Christensen, G.W. Euliss, M.J. McFadden, K.M. Coyle, P. Milojkovic, M.W. Haney, J. van der Gracht, and R.A. Athale. Active-eyes: an adaptive pixel-by-pixel image-segmentation sensor architecture for high-dynamic-range hyperspectral imaging. *Applied Optics*, 41(29):6093, October 2002. ISSN 0003-6935. doi: 10.1364/AO.41.006093.

Cura page. Web page of *Cura 12* program. `http://software.ultimaker.com/`, 2014.

M. Cuturi and A Doucet. Autoregressive kernels for time series. *arXiv preprint arXiv:1101.0673*, pages 1–21, 2011. URL `http://arxiv.org/abs/1101.0673`.

F.A.M. Davide, C. Di Natale, and A. D'Amico. Self-organizing multisensor systems for odour classification: internal categorization, adaptation and drift rejection. *Sensors and Actuators B: Chemical*, 18(1-3):244–258, March 1994. ISSN 09254005. doi: 10.1016/0925-4005(94)87090-X.

A. Depari, A. Flammini, D. Marioli, S. Rosa, and A. Taroni. A New Hardware Approach to Realize Low-Cost Electronic Noses. *Power*, pages 239–242, 2005.

A. Depari, P. Ferrari, A. Flammini, D. Marioli, S. Rosa, and A. Taroni. A New Modular Approach for Low-Cost Electronic Noses. *Interface*, 1(April): 578–583, 2006.

C. Di Natale, E. Martinelli, and A. D'Amico. Counteraction of environmental disturbances of electronic nose data by independent component analysis. *Sensors and Actuators B: Chemical*, 82(2-3):158–165, February 2002. ISSN 09254005. doi: 10.1016/S0925-4005(01)01001-2.

J. Ding, T.J. Mcavoy, R.E. Cavicchi, and S. Semancik. Surface state trapping models for SnO 2 -based microhotplate sensors. *Sensors (Peterborough, NH)*, 77:597–613, 2001.

C. Distante, P. Siciliano, and L. Vasanelli. Odor discrimination using adaptive resonance theory. *Sensors and Actuators B: Chemical*, 69(3):248–252, October 2000. ISSN 09254005. doi: 10.1016/S0925-4005(00)00502-5.

R. Dutta, E.L. Hines, J.W. Gardner, D.D. Udrea, and P. Boilot. Non-destructive egg freshness determination: an electronic nose based approach. *Measurement Science and Technology*, 14(2):190, 2003.

T.M. Dymerski, T.M. Chmiel, and W. Wardencki. Invited Review Article: An odor-sensing system-powerful technique for foodstuff studies. *Rev Sci Instrum*, 82(11):111101, November 2011. doi: 10.1063/1.3660805.

N. El Barbri, C. Duran, J. Brezmes, N. Cañellas, J.L. Ramírez, B. Bouchikhi, and E. Llobet. Selectivity Enhancement in Multisensor Systems Using Flow Modulation Techniques. *Sensors*, 8(11):7369–7379, November 2008. ISSN 1424-8220. doi: 10.3390/s8117369.

A.A. Ferreira, T.B. Ludermir, and R.R.B. de Aquino. A acomparative study of neural network to artificial noses. *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, 4:2081–2086, 2005.

J. Fonollosa, A. Vergara, and R. Huerta. Algorithmic Mitigation of Sensor Failure: Is Sensor Replacement Really Necessary? *Sensors and Actuators B: Chemical*, 183(211–221), March 2013. ISSN 09254005. doi: 10.1016/j.snb.2013.03.034.

Y. Freund and R.E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 904:23–37, 1995. doi: 10.1007/3-540-59119-2_166.

L. Gajdo. The concentration measurement with gas sensor operated in the dynamic regime. *Sensors and Actuators B: Chemical*, 106(2):691–699, May 2005. ISSN 09254005. doi: 10.1016/j.snb.2004.09.017.

C. García-Saura, F.B. Rodriguez, and P. Varona. Design Principles for Cooperative Robots with Uncertainity Aware and Resource Wise Adaptive Behavior. *Biomimetic and Biohybrid Systems*, 8608:108–117, 2014. doi: 10.1007/978-3-319-09435-9_10.

J.W. Gardner and P.N. Bartlett. A brief history of electronic noses. *Sens. Actuat. B: Chem.*, 18:211–220, 1994.

J.W Gardner and P.N. Bartlett. Electronic Noses. Principles and Applications. *Measurement Science and Technology*, 11(7):1087, 2000.

M.E. Gehm and J. Kinast. Adaptive spectroscopy: Towards adaptive spectral imaging. *Proceedings of SPIE*, 6978:69780I–69780I–11, April 2008. doi: 10.1117/12.779174.

T.D. Gibson, O. Prosser, J.N. Hulbert, R.W. Marshall, P. Corcoran, P. Lowery, E.A. Ruck-Keene, and S. Heron. Detection and simultaneous identification of microorganisms from headspace samples using an electronic nose., 1997. ISSN 09254005.

GNBot page. Web page of the GNBot created by Carlos García Saura. `https://github.com/carlosgs/GNBot`, 2013.

A.H. Gómez, J. Wang, G. Hu, and A.G. Pereira. Monitoring storage shelf life of tomato using electronic nose technique. *Journal of Food Engineering*, 85(4):625–631, April 2008. ISSN 02608774. doi: 10.1016/j.jfoodeng.2007.06.039.

R. Gosangi and R. Gutierrez-Osuna. Active temperature programming for metal-oxide chemoresistors. *Sensors Journal, IEEE*, 10(6):1075–1082, 2010.

R. Gutierrez-Osuna and A. Hierlemann. Adaptive microsensor systems. *Annual review of analytical chemistry (Palo Alto, Calif.)*, 3(February):255–76, January 2010. ISSN 1936-1335. doi: 10.1146/annurev.anchem.111808.073620.

F.G. Haibach and M.L. Myrick. Precision in multivariate optical computing. *Applied optics*, 43(10):2130–40, April 2004. ISSN 0003-6935.

C.W. Hanson and E.R. Thaler. Electronic nose prediction of a clinical pneumonia score: biosensors and microbes. Technical Report 1, Department of Anesthesia, University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA. hansonb@uphs.upenn.edu, 2005.

J.D. Hartman. A possible method for the rapid estimation of flavours in vegetables. *Proc. Amer. Soc. Hort. Sci.*, 64:335–342, 1954.

J.E. Haugen, E. Chanie, F. Westad, R. Jonsdottir, S. Bazzo, S. Labreche, P. Marcq, F. Lundby, and G. Olafsdottir. Rapid control of smoked Atlantic salmon (Salmo salar) quality by electronic nose: Correlation with classical evaluation methods. *Sensors and Actuators B: Chemical*, 116(1-2):72–77, July 2006. ISSN 09254005. doi: 10.1016/j.snb.2005.12.064.

A. Heilig, N. Bârsan, U. Weimar, M. Schweizer-Berberich, J.W. Gardner, and W. Göpel. Gas identification by modulating temperatures of SnO2-based thick film sensors. *Sensors and Actuators B: Chemical*, 43(1-3):45–51, September 1997. ISSN 09254005. doi: 10.1016/S0925-4005(97)00096-8.

F. Herrero-Carrón, D. Yañez, F.B. Rodriguez, and P. Varona. An active, inverse temperature modulation strategy for single sensor odorant classification. *Sensors and Actuators B: Chemical*, In press, 2014.

M. Holmberg, F. Winquist, I. Lundström, F. Davide, C. DiNatale, and A. D'Amico. Drift counteraction for an electronic nose. *Sensors and Actuators B: Chemical*, 36(1-3):528–535, October 1996. ISSN 09254005. doi: 10.1016/S0925-4005(97)80124-4.

X. Huang, Y. Choi, K. Yun, and E. Yoon. Oscillating behaviour of hazardous gas on tin oxide gas sensor: Fourier and wavelet transform analysis. *Sensors and Actuators B: Chemical*, 115(1):357–364, May 2006. ISSN 09254005. doi: 10.1016/j.snb.2005.09.022.

R. Huerta, S. Vembu, M.K. Muezzinoglu, and A. Vergara. Dynamical svm for time series classification. *Pattern Recognition*, 7476:216–225, 2012. doi: 10.1007/978-3-642-32717-9_22. URL `http://dx.doi.org/10.1007/978-3-642-32717-9_22`.

D. Hui, L. Jun-hua, and S. Zhong-ru. Drift reduction of gas sensor by wavelet and principal component analysis. *Sensors and Actuators B: Chemical*, 96 (1-2):354–363, November 2003. ISSN 09254005. doi: 10.1016/S0925-4005(03)00569-0.

K. Kato, Y. Kato, K. Takamatsu, T. Udaka, T. Nakahara, Y. Matsuura, and K. Yoshikawa. Toward the realization of an intelligent gas sensing system utilizing a non-linear dynamic response. *Sensors and Actuators B: Chemical*, 71(3):192–196, December 2000. ISSN 09254005. doi: 10.1016/S0925-4005(00)00604-3.

H.R. Keller and D.L. Massart. Peak purity control in liquid chromatography with photodiode-array detection by a fixed size moving window evolving factor analysis. *Analytica Chimica Acta*, 246(2):379–390, June 1991. ISSN 00032670. doi: 10.1016/S0003-2670(00)80976-9.

A.M. Kummer, T.P. Burg, and A. Hierlemann. Transient Signal Analysis Using Complementary Metal Oxide Semiconductor Capacitive Chemical Microsensors. *Analytical Chemistry*, 78(1):279–290, 2006. doi: 10.1021/ac051430g.

R. Kurnik. Application of the Mixtures of Experts algorithm for signal processing in a noninvasive glucose monitoring system. *Sensors and Actuators B: Chemical*, 60(1):19–26, November 1999. ISSN 09254005. doi: 10.1016/S0925-4005(99)00239-7.

A. Lee. Temperature modulation in semiconductor gas sensing. *Sensors and Actuators B: Chemical*, 60(1):35–42, November 1999. ISSN 09254005. doi: 10.1016/S0925-4005(99)00241-5.

C. Li, Y. Chen, M. Liu, and C. Lu. Utilizing diversified properties of monolayer protected gold nano-clusters to construct a hybrid sensor array for organic vapor detection. *Sensors and Actuators B: Chemical*, 169:349–359, July 2012. ISSN 09254005. doi: 10.1016/j.snb.2012.05.009.

A. Lilienthal and T. Duckett. Building gas concentration gridmaps with a mobile robot. *Robotics and Autonomous Systems*, 48(1):3–16, August 2004. ISSN 09218890. doi: 10.1016/j.robot.2004.05.002.

E. Llobet, E.L. Hines, J.W. Gardner, P.N. Bartlett, and T.T. Mottram. Fuzzy ARTMAP based electronic nose data analysis. *Sensors and Actuators B: Chemical*, 61(1-3):183–190, December 1999. ISSN 09254005. doi: 10.1016/S0925-4005(99)00288-9.

E. Llobet, J. Brezmes, R. Ionescu, X. Vilanova, and S. Al-khalifa. Wavelet transform and fuzzy ARTMAP-based pattern recognition for fast gas identi ® cation using a micro-hotplate gas sensor. *Sensors And Actuators*, 83:238–244, 2002.

A. Loutfi, S. Coradeschi, A. Lilienthal, and J. Gonzalez. Gas Distribution Mapping of Multiple Odour Sources using a Mobile Robot. *Robotica*, 27(2):311–319, 2008.

J. Lozano, M.J. Fernández, J.L. Fontecha, M. Aleixandre, J.P. Santos, I. Sayago, T. Arroyo, J.M. Cabellos, F.J. Gutiérrez, and M.C. Horrillo. Wine classification with a zinc oxide SAW sensor array. *Sensors and Actuators B: Chemical*, 120(1):166–171, December 2006. ISSN 09254005. doi: 10.1016/j.snb.2006.02.014.

J. Lozano, J.P. Santos, J. Gutiérrez, and M.C. Horrillo. Comparative study of sampling systems combined with gas sensors for wine discrimination. *Sensors and Actuators B: Chemical*, 126(2):616–623, October 2007. ISSN 09254005. doi: 10.1016/j.snb.2007.04.018.

C. Lytridis, E.E. Kadar, and G.S. Virk. A systematic approach to the problem of odour source localisation. *Autonomous Robots*, 20(3):261–276, June 2006. ISSN 0929-5593. doi: 10.1007/s10514-006-7414-3.

R.F. Machado, D. Laskowski, O. Deffenderfer, T. Burch, S. Zheng, P.J. Mazzone, T. Mekhail, C. Jennings, J.K. Stoller, J. Pyle, J. Duncan, R.A. Dweik, and S.C. Erzurum. Detection of lung cancer by sensor array analyses of exhaled breath. *American Journal of Respiratory and Critical Care Medicine*, 171(11):1286–1291, 2005.

S. Marco, A. Ortega, A. Pardo, and J. Samitier. Gas identification with tin oxide sensor array and self-organizing maps: adaptive correction of sensor drifts. *IEEE Transactions on Instrumentation and Measurement*, 47(1):316–321, 1998. ISSN 00189456. doi: 10.1109/19.728841.

J. Matthes, L. Gröll, and H.B. Keller. Source Localization by Spatially Distributed Electronic Noses for Advection and Diffusion. *Sensors (Peterborough, NH)*, 53(5):1711–1719, 2005.

J. Medendorp and R.A. Lodder. Applications of integrated sensing and processing in spectroscopic imaging and sensing. *Journal of Chemometrics*, 19(10):533–542, October 2005. ISSN 0886-9383. doi: 10.1002/cem.961.

M. Moens, A. Smet, B. Naudts, J. Verhoeven, M. Ieven, P. Jorens, H.J. Geise, and F. Blockhuys. Fast identification of ten clinically important microorganisms using an electronic nose. *Letters in applied microbiology*, 42(2):121–6, February 2006. ISSN 0266-8254. doi: 10.1111/j.1472-765X.2005.01822.x.

I. Morsi. Electronic Noses for Monitoring Environmental Pollution and Building Regression Model. *Science And Technology*, 2008.

T. Nakamoto, H. Matsushita, and N. Okazaki. Improvement of optimization algorithm in active gas/odor sensing system. *Sensors and Actuators A: Physical*, 50(3):191–196, September 1995. ISSN 09244247. doi: 10.1016/0924-4247(95)01039-4.

S. Nakata, S. Akakabe, M. Nakasuji, and K. Yoshikawa. Gas Sensing Based on a Nonlinear Response Discrimination between Hydrocarbons and Quantification of Individual Components in a Gas Mixture. *Analytical chemistry*, 68 (13):2067–72, July 1996. ISSN 0003-2700. doi: 10.1021/ac9510954.

S.K. Nayar and V. Branzoi. Adaptive dynamic range imaging: optical control of pixel exposures over space and time. *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1168–1175 vol.2, 2003. doi: 10.1109/ICCV.2003.1238624.

G. Ohloff. Smelling Materials and Sense of Smell, 1990.

Openscad page. Web page of *Openscad* program. `http://www.openscad.org/`, 2014.

L. Pan and S.X. Yang. Short Papers of Livestock Farm Odors. *Engineering*, 14 (3):371–376, 2009.

M. Pardo, L.G. Kwong, G. Sberveglieri, K. Brubaker, J.F. Schneider, W.R. Penrose, and J.R. Stetter. Data analysis for a hybrid sensor array. *Sensors and Actuators B: Chemical*, 106(1):136–143, April 2005. ISSN 09254005. doi: 10.1016/j.snb.2004.05.045.

A. Perera, T. Sundic, A. Pardo, R. Gutierrez-Osuna, and S. Marco. A portable electronic nose based on embedded PC technology and GNU/Linux: hardware, software and applications. *IEEE Sensors Journal*, 2(3):235–246, June 2002. ISSN 1530-437X. doi: 10.1109/JSEN.2002.800683.

A. Perera, N. Papamichail, N. Bârsan, U. Weimar, and S. Marco. On-Line Novelty Detection by Recursive Dynamic Principal Component Analysis and Gas Sensor Arrays Under Drift Conditions. *Sensors Journal, IEEE*, 6(3): 770–783, 2006.

K.C. Persaud and G. Dodd. Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose. *Nature*, 299:352–355, 1982.

D Pickel, G.P. Manucy, D.B. Walker, S.B. Hall, and J.C. Walker. Evidence for canine olfactory detection of melanoma. *Applied Animal Behaviour Science*, 89(1-2):107–116, 2004.

L. Pillonel, J.O. Bosset, and R. Tabacchi. Rapid Preconcentration and Enrichment Techniques for the Analysis of Food Volatile. A Review. *LWT - Food Science and Technology*, 35(1):1–14, February 2002. ISSN 00236438. doi: 10.1006/fstl.2001.0804.

R. Polikar, J. Byorick, and S. Krause. Learn++: a classifier independent incremental learning algorithm for supervised neural networks. *Neural Networks, ...*, 2002.

C.E. Priebe, D.J. Marchette, and D.M. Healy. Integrated sensing and processing decision trees. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):699–708, June 2004. ISSN 0162-8828. doi: 10.1109/TPAMI.2004.12.

Printrun page. Web page of *Printrun* program. `http://reprap.org/wiki/Printrun`, 2014.

P. Pyk, S. Bermúdez i Badia, U. Bernardet, P. Knüsel, M. Carlsson, J. Gu, E. Chanie, B.S. Hansson, T.C. Pearce, and P.F.M. J. Verschure. An artificial moth: Chemical source localization using a robot based neuronal model of moth optomotor anemotactic search. *Autonomous Robots*, 20(3):197–213, June 2006. ISSN 0929-5593. doi: 10.1007/s10514-006-7101-4.

B. Raman, D.C. Meier, J.K. Evju, and S. Semancik. Designing and optimizing microsensor arrays for recognizing chemical hazards in complex environments. *Sensors and Actuators B: Chemical*, 137(2):617–629, April 2009. ISSN 09254005. doi: 10.1016/j.snb.2008.11.053.

Reprap page. Web page of the printbot model. `http://reprap.org/wiki/Printrbot`, 2012a.

Reprap page. Web page of RepRap project. `http://reprap.org/`, 2012b.

R.R. Reston and E.S. Kolesar. Silicon-micromachined gas chromatography system used to separate and detect ammonia and nitrogen dioxide. I. Design, fabrication, and integration of the gas chromatography system. *Microelectromechanical Systems, Journal of*, 3(4):134–146, 1994. ISSN 1057-7157. doi: 10.1109/84.338634.

O.E. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5), 1976.

E. Schaller, B. Jacques O., and E. Felix. 'Electronic Noses' and Their Application to Food. *LWT - Food Science and Technology*, 31(4):305 – 316, 1998. ISSN 0023-6438. doi: http://dx.doi.org/10.1006/fstl.1998.0376.

K.D. Schierbaum, U. Weimar, and W. Göpel. Multicomponent gas analysis: An analytical chemistry approach applied to modified SnO2 sensors. *Sensors and Actuators B: Chemical*, 2(1):71–78, March 1990. ISSN 09254005. doi: 10.1016/0925-4005(90)80011-N.

SDCC page. Web of SDCC compiller. `http://sdcc.sourceforge.net/`, 2007.

K.K. Shukla, R.R. Das, and R. Dwivedi. Adaptive resonance neural classifier for identification of gases/odours using an integrated sensor array. *Sensors and Actuators B: Chemical*, 50(3):194–203, August 1998. ISSN 09254005. doi: 10.1016/S0925-4005(98)00236-6.

U. Siripatrawan. Self-organizing algorithm for classification of packaged fresh vegetable potentially contaminated with foodborne pathogens. *Sensors and Actuators B: Chemical*, 128(2):435–441, January 2008. ISSN 09254005. doi: 10.1016/j.snb.2007.06.030.

Skybot wiki page. Web of the PIC-Bootloader. `http://www.iearobotics.com/wiki/index.php?title=PIC_Bootloader`, 2007a.

Skybot wiki page. Web to use Pydownloader. `http://www.iearobotics.com/wiki/index.php?title=Pydownloader-wx`, 2007b.

Skybot wiki page. Web to use SkyBot-Test. `http://www.iearobotics.com/wiki/index.php?title=Skybot-test`, 2007c.

Skybot wiki page. Web to install programs to use Skybot in Linux. `http://www.iearobotics.com/wiki/index.php?title=Skypic:Software_Linux`, 2007d.

Skybot wiki page. Web to install programs to use Skybot in Windows. `http://www.iearobotics.com/wiki/index.php?title=Pydownloader-wx#Instalaci.C3.B3n_en_Windows`, 2007e.

Skypic wiki page. Sky293 web page. `http://www.iearobotics.com/proyectos/sky293/sky293.html`, 2007a.

Skypic wiki page. Skypic wiki page. `http://www.iearobotics.com/wiki/index.php?title=Skypic`, 2007b.

J.H. Sohn, M. Atzeni, L. Zeller, and G. Pioggia. Characterisation of humidity dependence of a metal oxide semiconductor sensor array using partial least squares. *Sensors and Actuators B: Chemical*, 131(1):230–235, April 2008. ISSN 09254005. doi: 10.1016/j.snb.2007.11.009.

K. Song, Q. Liu, and Q. Wang. Olfaction and hearing based mobile robot navigation for odor/sound source search. *Sensors (Basel, Switzerland)*, 11 (2):2129–54, January 2011. ISSN 1424-8220. doi: 10.3390/s110202129.

C. Stachniss, P. Christian, L. Achim, and B. Wolfram. Gas Distribution Modeling using Sparse Gaussian Process Mixture Models. *Advanced Robotics*, 2008.

Steven H., S. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Perseus Books Group, 2001. ISBN 0738204536.

S.E. Stitzel, M.J. Aernecke, and D.R. Walt. Artificial noses. *Annu Rev Biomed Eng*, 13:1–25, 2011. doi: 10.1146/annurev-bioeng-071910-124633.

T.B. Tang, H. Chen, and A.F. Murray. Adaptive, integrated sensor processing to compensate for drift and uncertainty: a stochastic 'neural'approach. *IEE Proceedings-Nanobiotechnology*, 151(1):28–34, 2004. doi: 10.1049/ip-nbt.

H. Ulmer, J. Mitrovics, G. Noetzel, U. Weimar, and W. Göpel. Odours and flavours identified with hybrid modular sensor systems. *Sensors and Actuators B: Chemical*, 43(1-3):24–33, September 1997. ISSN 09254005. doi: 10.1016/S0925-4005(97)00161-5.

S. Vaihinger, W. Göpel, and J.R. Stetter. Detection of halogenated and other hydrocarbons in air: Response functions of catalyst/electrochemical sensor systems. *Sensors and Actuators B: Chemical*, 4(3-4):337–343, June 1991. ISSN 09254005. doi: 10.1016/0925-4005(91)80133-5.

S. Vembu, A. Vergara, M.K. Muezzinoglu, and R. Huerta. On time series features and kernels for machine olfaction. *Sensors and Actuators B: Chemical*, 174:535–546, November 2012. ISSN 09254005. doi: 10.1016/j.snb.2012.06.070.

A. Vergara, E. Llobet, J. Brezmes, P. Ivanov, X. Vilanova, I. Gracia, C. Cané, and X. Correig. Optimised temperature modulation of metal oxide microhotplate gas sensors through multilevel pseudo random sequences. *Sensors and Actuators B: Chemical*, 111:271 – 280, 2005. ISSN 0925-4005. doi: http://dx.doi.org/10.1016/j.snb.2005.06.039.

J.S. Vestergaard, M. Martens, and P. Turkki. Application of an electronic nose system for prediction of sensory quality changes of a meat product (pizza topping) during storage. *LWT - Food Science and Technology*, 40(6):1095–1101, August 2007. ISSN 00236438. doi: 10.1016/j.lwt.2006.06.008.

S. Waphare, D. Gharpure, A. Shaligram, and B. Botre. Implementation of 3-Nose Strategy in Odor Plume-Tracking Algorithm. *2010 International Conference on Signal Acquisition and Processing*, pages 337–341, February 2010. doi: 10.1109/ICSAP.2010.81.

G. Wei, Z. Tang, P.C.H. Chan, and J. Yu. A Blind Source Separation Based Micro Gas Sensor Array Modeling Method. *Advances in Neural Networks – ISNN 2004*, 3173:696–701, 2004.

B.M. Wilamowski and V.J. Vodyanoy. Neural network architectures for artificial noses. *2008 Conference on Human System Interactions*, pages 731–736, May 2008. doi: 10.1109/HSI.2008.4581532.

A.D. Wilson and M. Baietto. Applications and Advances in Electronic-Nose Technologies. *Sensors*, 9(7):5099–5148, June 2009. ISSN 1424-8220. doi: 10.3390/s90705099.

H. Xu, M. Zhang, H. Ding, and Z. Xie. Colloidal silica beads modified with quantum dots and zinc (II) tetraphenylporphyrin for colorimetric sensing of ammonia. *Microchimica Acta*, 180(1-2):85–91, 2013. ISSN 0026-3672. doi: 10.1007/s00604-012-0914-2.

D.J. Yáñez, A. Toledano, E. Serrano, A.M. Martín de Rosales, F.B. Rodríguez, and P. Varona. Characterization of a clinical olfactory test with an artificial nose. *Frontiers in neuroengineering*, 5(February):1, October 2011. ISSN 1662-6443. doi: 10.3389/fneng.2012.00001.

M. Zuppa. Drift counteraction with multiple self-organising maps for an electronic nose. *Sensors and Actuators B: Chemical*, 98(2-3):305–317, March 2004. ISSN 09254005. doi: 10.1016/j.snb.2003.10.029.

H. Zwaardemaker and Hogewind F. On spray-electricity and waterfall-electricity. *In KNAW Proceedings, Amsterdam, Netherlands*, 22:429–437, 1920.